

ATTN: FILE 0023

AD-A214 262



DYNAMIC ARCHITECTURE COMPUTER

THESIS

Patrick E. Price

AFIT/GE/ENG/89D-40

This document has been approved  
for public release and sale in  
distribution to individuals.

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DTIC  
ELECTE  
NOV 13 1989  
S E

20 7 7 7 0 0 10

AFIT/GE/ENG/89D-40

*D*

DYNAMIC ARCHITECTURE COMPUTER

THESIS

Patrick E. Price

AFIT/GE/ENG/89D-40

**DTIC**  
**ELECTE**  
**NOV 13 1989**  
**S E D**

Approved for public release; distribution unlimited

AFIT/GE/ENG/89D-40

DYNAMIC ARCHITECTURE COMPUTER

THESIS

Presented to the Faculty of the School of Engineering  
of the Air Force Institute of Tecnology  
Air University

In Partial Fulfillment of  
Master of Science in Electrical Engineering

Patrick E. Price, B.S.

December 1988

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

Approved for public release; distribution unlimited

## Preface

The purpose of this thesis was to design a computer that could process a large variety of calculations with a minimum of hardware. This constraint requires a computer that can change its structure to match the demands of the problem currently being calculated. Computer image generation was selected as an example problem. The processing requirements of real-time computer image generation require calculation of very large real numbers as well as very small logical variables.

The results demonstrate that, in a best case analysis, a dynamic architecture computer can demonstrate an improvement in processing speed over conventional single instruction, single data computers. JPS

In preparing this thesis, I extend my gratitude to several people for their contributions. First and foremost, I thank my advisor, Dr. Thomas C. Hartrum, for his guidance and support. Also, I thank Captain Nathaniel Davis IV and Captain Bruce George for their expertise and assistance. Finally, I thank Ms. Deborah Martin for her help in tabulating the statistics.

Patrick E. Price

## Table of Contents

Preface .....	ii
List of Figures .....	v
List of Tables .....	vi
I. Introduction .....	1
Background .....	1
Problem .....	3
Scope .....	4
Approach .....	4
Thesis Organization .....	5
II. Literature Review .....	7
Introduction .....	7
Estrin .....	9
Kartashev and Kartashev .....	12
Dimond and King .....	38
Rauscher and Agrawala .....	39
Fuchs and Johnson .....	40
III. Computer Generated Imagery Software .....	42
Computer Generated Images .....	42
Scene Generation Software .....	44
Required Data .....	51
Software Analysis .....	54
IV. Dynamic Architecture Computer Design .....	68
Dynamic Computer Features .....	77
Dynamic Computer Operation .....	86
Summary .....	92
V. Analysis and Conclusions .....	93
Analysis .....	93
Conclusions .....	101
Recommendations for Further Research ...	103

References .....	109
Bibliography .....	111
Appendix A: Module Calling Summary .....	112
Appendix B: Module Descriptions .....	125
Appendix C: Collected Data on Variables .....	136
Appendix D: Collected Data on Operations .....	208

## List of Figures

<u>Figure</u>		<u>Page</u>
2-1	Keyword Search Strategy .....	8
2-2	Flowchart for Constructing a Program Graph .	31
3-1	Data Base Development System Diagram .....	45
3-2	Coordinate Set Definition .....	48
4-1	Format of Floating Point Variables .....	76
4-2	Possible Configurations .....	77
4-3	Instruction Format .....	84
4-4	Carry-In/Carry-Out Structure .....	85
4-5	Dynamic Architecture Computer .....	87
5-1	Percent of Execution Time Required by Dynamic Computer .....	100

## List of Tables

<u>Table</u>		<u>Page</u>
3-1	Valid Variable Type and Size Combinations ..	58
3-2	Summary of Variable Data .....	60
3-3	Total Variable Data .....	61
3-4	Itemization of Variable Operations Data ....	65
3-5	Summary of Variable Operations Data .....	65
3-6	Summary of Other Operations Data .....	66
4-1	Relative Occurances by Variable Type .....	73
4-2	SEL 32/70 Instruction Repertoire .....	75
4-3	Summary of Variable Addressing Requirements	83
5-1	Summary of Processors and Operations .....	98
5-2	Summary of Configurations and Operations ...	98
5-3	Detailed Execution Analysis .....	98

# DYNAMIC ARCHITECTURE COMPUTER

## I. Introduction

### Background

Digital computers may be designed and built using discrete components, individual integrated circuits or microcomputer chips. A variation of the microcomputer chip is the bit-slice chip. Each bit-slice chip contains all of the circuits and components that would be obtained by slicing through the processing portion of a computer. Thus, each slice could become a small stand-alone computer if properly connected to memory and other peripheral devices. Large computers may be built by using a number of these slices connected together. It is also possible to use these bit-slice chips to build a computer that is very fast for a specific application.

This is a desirable concept because general purpose computers are not fast enough for certain applications. One example of particular interest is Computer Generated Imagery (CGI). CGI requires a data base of digitized descriptions of three-dimensional features. By careful manipulation of these descriptions, a realistic visual scene is created that can be viewed on a television picture tube. This requires a substantial number of calculations in order to create the proper perspective and size of each object and to convert each object to individual picture elements for display on a two dimensional screen. Furthermore, if the illusion of motion is to be created, these calculations must be done at least 30 times per second.

A general purpose computer is designed to handle a variety of tasks equally well. Applications like CGI require that the hardware be highly tuned for several specific types of data manipulations. Therefore, the computation of CGI algorithms is generally done in special purpose processors. The CGI algorithms are implemented directly in the hardware of these special purpose processors. If the CGI algorithms change for any reason, it is not possible to change the special purpose processor without a redesign of the hardware.

Currently, there is some interest in developing general purpose digital computers that can vary their architecture dynamically. That is, they can change from a computer that handles large, high precision numbers into a computer that handles smaller, less precise numbers. When this computer is processing smaller numbers, it would be able to do several calculations in parallel.

Bit-slice chips make ideal building blocks for a dynamic architecture computer, and CGI is a very good application for testing such a design. A general purpose dynamic architecture computer would be very complex because it would have to be able to assume all possible combinations of connections. A dynamic architecture computer designed to perform CGI could be simplified to perform only those operations essential to CGI.

### Problem

The problem is to design a dynamic architecture computer for the specific purpose of processing Computer Generated Imagery (CGI) algorithms and to demonstrate that a savings in time can be achieved by using this computer instead of a general purpose computer of fixed architecture.

## Scope

This effort includes a design for the computer in sufficient detail to make accurate timing calculations. For purposes of this effort, the design will not be taken to the level that an actual machine could be constructed, although there will be recommendations for implementing a prototype and data for a prototype test. Whenever possible, the design will be such that it could be extended to a general purpose computer if desired as a follow-on effort.

## Approach

The general procedure followed during the conduct of this study was as follows:

Literature Review. The literature review concentrated on researching the work already done in the area of dynamic computer architecture. The review assured that this study did not duplicate previous studies and provided the background information for this study.

Analysis of CGI. This phase concentrated on analyzing a software emulation of some typical CGI hardware. The results of this analysis consisted of details of required instruction sets and the size and precision of the variables being calculated.

Design of Architecture. The result of this phase was the design of a dynamic architecture computer based on the information obtained from the analysis outlined in the steps above.

Analysis of Results. This concluding phase determined whether or not the resulting design demonstrated an improvement in speed over a fixed architecture computer performing the same task.

### Thesis Organization

The organization of this thesis follows the steps outlined in the approach. Chapter One contains the background and other introductory material. Chapter Two contains a review of the pertinent literature. Chapter Three discusses the analysis of the Computer Generated

Imagery (CGI) software including the organization and operation of the actual software, the type of data desired as a result of the analysis, and the steps performed in doing the analysis. Chapter Four details the design of the architecture. Chapter Five contains the analysis of the results.

## II. Literature Review

### Introduction

A literature review was undertaken as the first step in this research. The purpose of the review was to find those articles published on the general topic of dynamic computer architecture. The results and a discussion of the most important items found are given below.

The primary literature search into dynamic computer architectures was performed using the Lockheed automated data retrieval system to do a keyword search on the COMPENDEX (Corporate Engineering Index Inc.) file. Also at this time, a search was performed on both INSPEC files and the NITS file using the same search strategy. Of the abstracts obtained in this manner, only a few were directly related to dynamic computer architecture. The keyword search strategy is given by Figure 2-1.

Each relevant article is discussed below. The discussions are arranged chronologically by author.

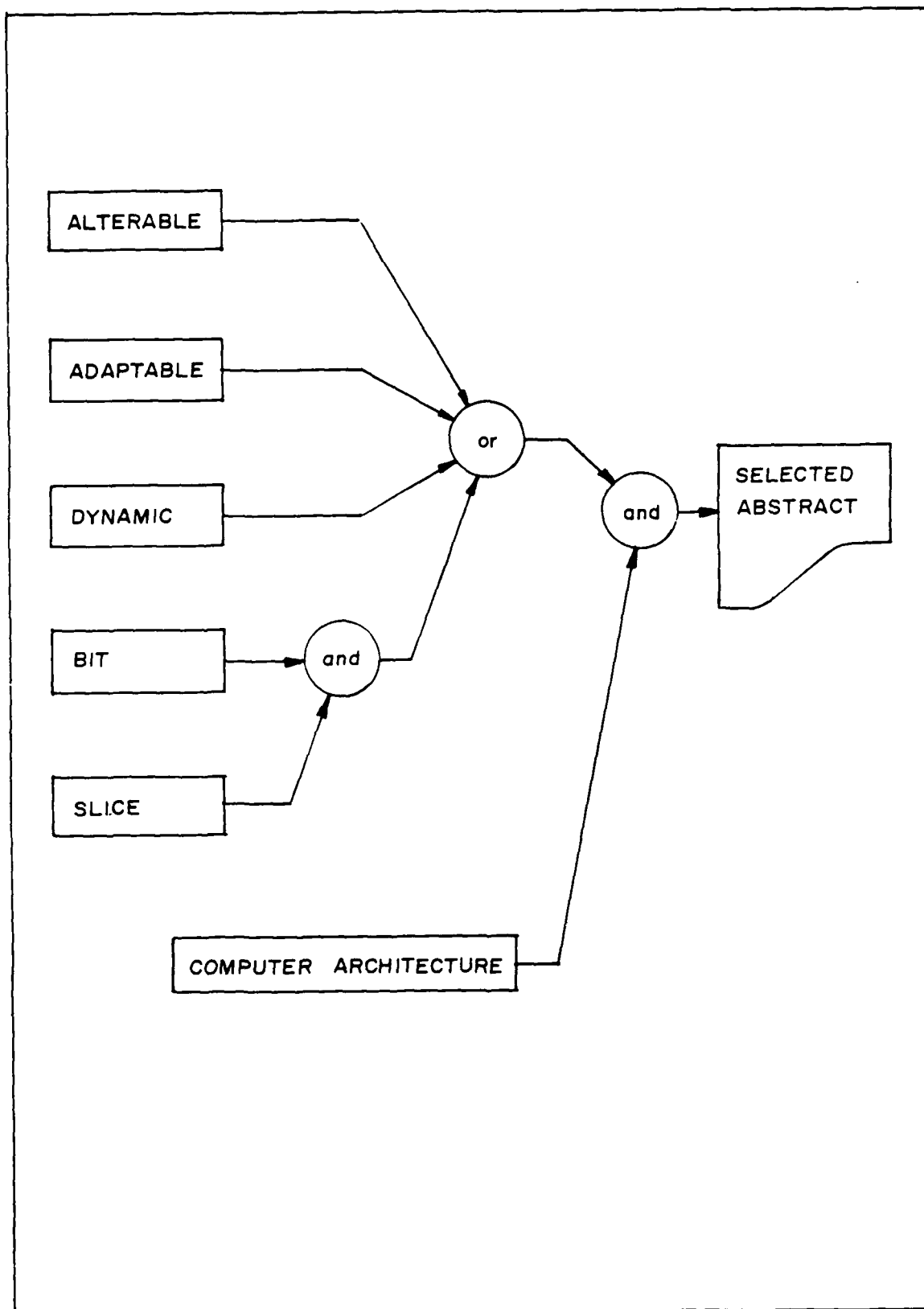


Figure 2-1: Keyword Search Strategy

## Estrin

The first paper chronologically is "Organization of Computer Systems: The Fixed Plus Variable Structure Computer." This article describes a proposed computer system that could be adapted to specific problems. This computer system would be composed of two parts. The first part would be fixed in its architecture. The fixed part would actually be an off the shelf general purpose computer.[Ref 2:34]

The second part of this system would be variable both in terms of the individual components used and in the interconnection of the components. Each of these separate components could be any of the fundamental elements of a regular computer such as flip-flops or shift registers. In addition to the individual elements, there would be a library of frequently used substructures that are hardwired combinations of individual elements. Each of the individual elements and substructures could be connected together in a variety of ways within the variable portion of the system.[Ref 2:34-35]

The significance of this paper is that it is the first to mention variable connections between elementary circuits within a computer. The computer system described in this paper is apparently not dynamic, but the basic ideas for a variable architecture computer are expressed.

The next paper reviewed was "Parallel Processing in a Restructurable Computer System." This article describes the Fixed Plus Variable computer system as it was being built at UCLA.[Ref 3:747-755]

In addition to the fixed and variable portions described above, a supervisory control unit has been added to the design. This control unit is built in several levels where each level exercises control over a certain type of operation. For instance, the lowest level executes the lowest level single action such as arithmetic and logical operations. The next higher level executes the elementary functions such as complex arithmetic and matrix operations. This level uses all of the operations defined at the lowest level in order to perform its functions.[Ref 3:749-750]

Intermediate levels may exist which would execute higher level functions. Each level defines its own functions using the functions defined at all levels beneath it.[Ref 3:750]

The highest level of control is the supervisory level. This level performs the following special functions:

1. Controls execution of all computations in the fixed and variable portions
2. Coordinates the information exchange between the fixed and variable portions
3. Performs interlocking functions necessary for parallel processing

[Ref 3:750]

In addition, the control units on each level have the following functions:

1. Sequence through each state required to perform the necessary operation.
2. For each state, perform the necessary commands, using the functions defined at the lower levels.
3. Generate the next sequential state.

[Ref 3:750-751]

Kartashev and Kartashev

The most important articles found were a series of articles written by Steven I. Kartashev and Svetlana P. Kartashev. Over a period of several months they have published the details of their design for a dynamic computer architecture that has many of the attributes that this thesis is trying to achieve. The content of some of the articles overlaps so only the ones required for an all inclusive design are included here. The first of these articles is "A Powerful LSI Metacomputer System with Dynamic Architecture for Simulation of Complex Problems." [Ref 5:483-488] It discusses a dynamic computer architecture that could be used to speed up the calculations for distributing electric power from a grid of power generating plants. [Ref 5:488]

The dynamic architecture computer described in this article is composed of a number of "dynamic computer groups." Each group is identical to every other group in that they all contain the same elements. Each group can function independently or in conjunction with other groups to form a larger group. [Ref 5:483-484]

In addition to each group being identical, each individual group contains, within itself, a number of identical sets of elements. Each of these elements consists of a processor unit, a memory unit and an input/output unit. Each element can therefore be thought of as a simple computer. These elements can also function independently or in conjunction with other elements in its own group. The elements within a group are connected together so the data can be passed either left and right between the elements or only between the memory and its associated processing unit. Each group has a monitoring unit, called a V monitor, which controls the interconnections between the individual elements.[Ref 5:483-486]

The simple computers in this case are all 16 bits wide and have a 16 bit wide memory. These simple computers may be dynamically linked together through the connecting units to form wider computers in multiples of 16 bits. Each computer can process data concurrently with the computers formed by the other elements.[Ref 5:485-486]

Obviously an arrangement such as this could make the memory access quite complicated. This design solves some of the problems by making the memory access both serial and

parallel. When two elements are connected together to form one 32 bit computer, each 16 bit portion accesses the same memory location in its 16 bit memory. The connecting unit described above is in the no pass mode so that the data goes from the memory to its associated processor unit.[Ref 5:484]

Instructions, however, are stored in consecutive locations within one memory unit. When an instruction is obtained from a location in one of the memory units, it is passed either left or right through the connecting unit to all of the affected processing units. A single program may be stored in more than one memory unit. Execution control is passed to the instruction stream of the next memory by a special jump instruction.[Ref 5:485]

The article "Designing LSI Modular Computers and Systems" [Ref 6:1-6] elaborates on this basic design by discussing the V monitor in further detail, by discussing the principles of design of the operating system and by introducing the concept of program universality.[Ref 6:1]

The V monitor is the control unit for each dynamic computer group. It controls the transition between states and resolves conflicts of requests for new configurations by

the programs. Task execution within each element is concluded by a STOP instruction which informs the monitor that that resource is now free and available for reassignment.[Ref 6:5-7]

The operating system is composed of three basic programs. The assignment portion is the first to see the user program. It breaks the user program into segments of known bit size, organizes these segments into tasks of common bit size, and then assigns the hardware resources needed to run each task. The second portion of the operating system is the local monitor which runs in the V monitor of each group. Its functions have already been described. The third portion is the central monitor program which runs in the system's control computer.[Ref 6:8-9]

The central monitor program manages the resources of the entire system. It acts much like the local monitor does only on a system wide basis. Its tasks are to prioritize all requests for transition of the entire system, keep track of and specify each group's ability to transition into new groups, and it interrupts lower priority programs and obtains the necessary resources for higher priority programs.[Ref 6:9]

The third important item in this article is the introduction of the concept of program universality. This concept is essential to the capability to perform multiple architecture switches. The principal concepts are:

- (1) Instructions store no codes or constants which change their meanings when the same program is computed by different size computers
- (2) Instruction size is unique and independent of computer size
- (3) Addresses in the instruction fields remain unchanged when moving programs from computer to computer

[Ref 6:9]

The functioning of the operating system is elaborated further in the article "Dynamic Architectures: Problems and Solutions." [Ref 7:26-40] Any operating system for dynamic architectures must have two additional functions:

- (1) It must construct a diagram of the computer sizes needed
- (2) It must flowchart the architectural states and assign a priority for the transitions

The operating system, as conceived in this article, contains additional subsystems to accomplish this task. Previous

articles divided the monitor system into three parts. This article combines the functions of the monitor system into one subsystem.[Ref 7:35]

The basic tasks of the assignment subsystem were described above. That is, it takes the source code and organizes it into specific bit sized pieces and inserts the transition instructions. This process is done in four steps as follows:

1. Break the source statements into nodes where the beginning and end of each node occurs at a control statement.

2. Find the maximum bit size of all computations. Algorithms are given in this article to find these maximums.

3. Use the maximum bit size of the computations in the node to establish the maximum size of the computer needed to execute that node.

4. A two axis diagram of bit sizes is built. The horizontal axis represents the number of graph nodes and the vertical axis represents the computer bit size for each node.[Ref 7:35-38]

The next article in the series, "LSI Modular Computers, Systems, and Networks" [Ref 8:7-15] is an

introduction to a special issue of "Computer" magazine published by IEEE. The importance of this article to this discussion is the definition of the terms Static Architecture, Dynamic Architecture, and Reconfigurable Architecture.[Ref 8:7-9]

Static architecture allows no software controlled variations. Reconfigurable architecture allows partially software controlled variations in the module's interconnections. Dynamic architecture allows complete software controlled reconfiguration.[Ref 8:9]

"Software Problems for Dynamic Architectures: Adaptive Assignment of Hardware Resources" [Ref 9:775-780] expands the discussion of functions of the assignment subsystem. The four steps mentioned earlier as being done by the assignment subsystem are now looked at from a different perspective. These tasks are divided into three topics and discussed in detail. These three topics are construction of a program graph, diagram of hardware resources, and assignment of the DC group resource among programs.[Ref 9:775]

The basic unit of construction of a program graph is the node. The construction of a node was discussed previously as being all of the statements between two consecutive control points. Control points are statements where program execution forks or joins. The maximum bit size of each node is calculated by the following procedure:

1. Each variable is analyzed to determine its maximum bit size
2. Each statement is analyzed to determine the maximum bit size required for its calculations
3. Each calculation is analyzed to determine the maximum intermediate bit size required. The maximum intermediate bit size is that size required to contain the intermediate results.

Formulas are given in the article for determining the maximum bit size and the maximum intermediate bit size for various arithmetic expressions.[Ref 9:775-777]

Once the program graph is constructed, the diagram of hardware resources can be made. This is a four step process which uses the data derived by the procedures described above. The first step is to construct the bit size diagram. The horizontal axis of this diagram represents the nodes of the program graph. The vertical axis shows the two

bit size parameters, the maximum bit size and the maximum intermediate bit size.[Ref 9:777]

The next step is to adjust the bit size diagram to eliminate excessive changes in computer sizes. The result of this is a computer size diagram that is ordered in the sequence of computer sizes required.[Ref 9:777]

The third step is to determine the time required to execute each task in its given computer size. This can be done by breaking down each statement into its machine code equivalent. The number of clock periods for each machine instruction is based on the computer size and memory access speed. This value is multiplied by the number of times it is iterated to find the total time for that instruction. The total time for all instructions in that node are added together to get the total time for that node.[Ref 9:777]

The fourth step is to construct the hardware resource diagram for the entire program. This is called the P-resource diagram. It is a graph where the time of functioning of each task is plotted on the horizontal axis. The upper portion of the vertical axis is plotted with the computer sizes. The lower portion is plotted with the dimensions of the data arrays.[Ref 9:777-778]

The third topic of this article is the assignment of the DC group resources among the programs. This is accomplished by combining all the P-resource diagrams (output from stage 4 of the hardware resource diagram) into Computational Element (CE) resource diagrams and the Memory Element (ME) resource diagrams.[Ref 9:778,780]

The CE resource diagram plots the maximum bit size on the vertical axis and the time for executing each task is mapped along the horizontal axis. The construction of this diagram is done in accordance with the program priorities. That is, the high priority program segments are plotted first. The result is a diagram that maps all of the computer resource requirements for the programs to be executed.[Ref 9:778-779]

The ME resource diagram is built using the memory size portions of all P-resource diagrams. All data arrays are assigned first because they must use the same location in all memories. The remaining spaces are filled in with programs and program segments since execution can jump from memory to memory. The result is a graphic picture of the memory space required to execute the subject programs.[Ref 9:779-780]

The concept of dynamic architecture is extended to pipeline systems in the article "Adaptable Pipeline System with Dynamic Architecture." [Ref 10:222-230] This article proposes a design for a pipeline computer system that uses the same Dynamic Computer (DC) groups that were presented in previous articles. Each stage of the pipeline is made from a single DC-group. In addition, each stage has its own register set for storage of temporary results. [Ref 10:222,225]

This dynamic pipeline architecture solves some of the problems of ordinary pipeline architectures by allowing the instruction to exit the pipeline when execution is completed even though more stages remain in the pipeline. Also, each stage has a variable execution time for each instruction being executed. [Ref 10:224]

The next paper in this series, "Adaptation Properties for Dynamic Architectures," [Ref 13:543-556] introduces a concept called adaptation parameters. These parameters allow the user program to be evaluated against alternative architectures. These evaluations will select the optimum architecture for each program. Equations and examples for each calculation are given in the article. [Ref 13:543-556]

The first of these parameters is the Speed of Bit size Adaptation (SBA). This parameter is the time that it takes the computer to switch from one architectural state to another. This number is a factor of the switching configuration and the technology used to implement the switch.[Ref 13:544-545]

The second parameter is the Precision of Bit size Adaptation (PBA). This parameter represents the time lost in executing instructions in a machine size too large for that particular instruction. Each instruction is likely to require a different size computer. Therefore, in order to minimize the switches between states, instructions are grouped into tasks of similar computer length and these tasks are then executed in a fixed computer size. However, there will still be instances where an instruction, within a given task, could have been executed in a smaller computer. The difference in time between its execution in its assigned task and the time it would have taken to execute in a smaller machine is a loss of efficiency. The sum of all of these losses throughout a program is the PBA.[Ref 13:545]

The next parameter is called the Resource Utilization Factor (RUF). This parameter is computed for each state

that the system is in during the execution of a program. In each state, the system assumes a number of computer sizes that execute concurrent instruction streams. When a task is executed in this state, some of the processes finish before others and the resources of that path must be idle. This idle time is used to calculate the RUF.[Ref 13:545-546]

In addition to dynamic architecture, it is also conceivable that instruction sets can be dynamically changed. The difference in execution time of one instruction set over another for the same program is a parameter called Speed-up on Program Adaptation (SPA). Related to this is a parameter that computes the gain in speed obtained by implementing an often performed instruction stream into a single executable instruction. This parameter is called the Speed-up by Instruction Adaptation (SIA).[Ref 13:546-547]

There is a parameter that measures the efficiency with which a dynamic architecture adapts to array processing. This factor is called the Array Adaptation of Equipment (AAE). It is the percentage of equipment left over when a computer size is selected that is larger than the operands. It is similar to the factor PBA but it is specifically for array structures.[Ref 13:547-548]

There are numerous factors which must be considered in adapting a dynamic architecture to a pipeline configuration:

Adaptation to parallel streams

Adaptation on operation sequences

Adaptation on the number of pipeline stages

Adaptation to operation time in each stage

Adaptation on conditional branch

These factors are also described in this article but since they deal with pipeline architecture, there is no need to elaborate on them here.[Ref 13:548-550]

The final parameter is the time that it takes to adapt a program so that it may be executed. This is called the Time of Program Adaptation (TPA). The ideal situation is a TPA of zero or no time required to adapt the program. This occurs with all programs constructed under the rules of program universality. Program universality was presented in detail above. Its important points are:

- 1) all instruction codes have the same meaning regardless of the computer size
- 2) unique instruction size
- 3) serial consecutive storage of instructions in memory

4) parallel storage of data in memories.

However, complete program universality is not always practical. Therefore, a certain amount of time is usually required in order to adapt a given program to a new architecture. This time is called the TPA.[Ref 13:550]

This article also departs from the previously defined operating system by adding an additional system. The adaptation system is now the first system to process the user's program. Its job is to find the optimum instruction set for executing this program. The other two portions of the operating system, the assignment and the monitor, remain the same.[Ref 13:550]

The article "A Multicomputer System With Dynamic Architecture" [Ref 11:704-721] includes more detail about the function of the monitor system. Specifically, it deals with those things which must be done in performing the switch from one architectural state to another. The previous articles divided the monitor system into sections based on where in the computer system each portion was located (i.e. local monitor, V-monitor, etc.). This article discusses the monitor system in functional areas. They are:

- 1) Task synchronization

- 2) Priority analysis
- 3) Storage of variable control codes
- 4) Organization of the architectural switch to a new state.[Ref 11:706,715]

The first two functions are self explanatory. They are performed by the V-monitor during execution of programs in the dynamic computer. These two processes handle the reallocation of resources in real-time. Task synchronization determines when the resources of a particular CE are free and ready to be transitioned. Priority analysis is required in order to determine which tasks or programs will get the available resources for its processing.[Ref 11:715]

Storage of variable control codes is done by the Central Monitor each time a new DC group is formed. These variables are written into each individual CE's memory in order to switch the architecture to a new state. The control codes for all CE's for all possible configurations are stored in one of the memories where they can be accessed by the appropriate V-monitors. These codes are used by the system to denote the current configuration and so each CE knows how it is supposed to be configured.[Ref 11:715-717]

The next article "Adaptable Architectures for Supersystems" refines the details of the monitor system. Its discussion of the monitor system divides the function into four different actions. They are:

1)Checking the readiness of the resources requested for reconfiguration

2)Task synchronization

3)Priority analysis

4)Architectural reconfiguration.[Ref 15:34-35]

The total monitor system operation as described in this article is not different from the previously presented concepts. However, there are more details given on the implementation of these functions.[Ref 15:34-35]

The final article of interest by these authors is "Distribution of Programs for a System with Dynamic Architecture." It is important mainly because of its detailed presentation of an algorithm for constructing a program graph.[Ref 12:490-492]

A program graph consists of a series of nodes, connected by execution flow lines. Nodes can be simple or complex and may also be iterative or non-iterative. Simple

nodes have only one exit point for control to pass to the next node. Complex nodes contain some type of decision statement and therefore have more than one exit point. For a complex node, the particular node to which control passes next is determined by some type of decision statement internal to that node. All decision statements are considered control statements.[Ref 12:489]

Iterative nodes are executed some number of times specified by a parameter called Z. Non-iterative nodes are executed only once in the course of execution of that particular program path.[Ref 12:490]

An important part of the algorithm which does not become a part of the graph is the node cross reference table. This is a two column table that is used to keep track of which nodes need to be connected at a later time to other nodes. Column one contains a pointer to the control statements. Column two contains pointers to all of the statements being referenced by the control statement.[Ref 12:490]

The algorithm divides all of the statements in a user's program into five types as follows:

Type 1: A non-control statement that is not referenced by any other statement.

Type 2: A statement referenced by another control statement

Type 3: All control statements except the DO statement

Type 4: The DO statement

Type 5: The DO reference statement or the DO object.[Ref 12:490-492]

This algorithm is illustrated by the flowchart in Figure 2-2. The result of using this algorithm on a program is a flow graph that shows all of the executable program statements and all of the possible execution paths.[Ref 12:491]

Type 1 statements become part of the previous node unless the previous node contains a control statement. If the preceding node is a control statement node, then the type 1 statement in question is made into a new separate node. Consecutive type 1 statements are collected together by the algorithm into a single simple node.[Ref 12:490]

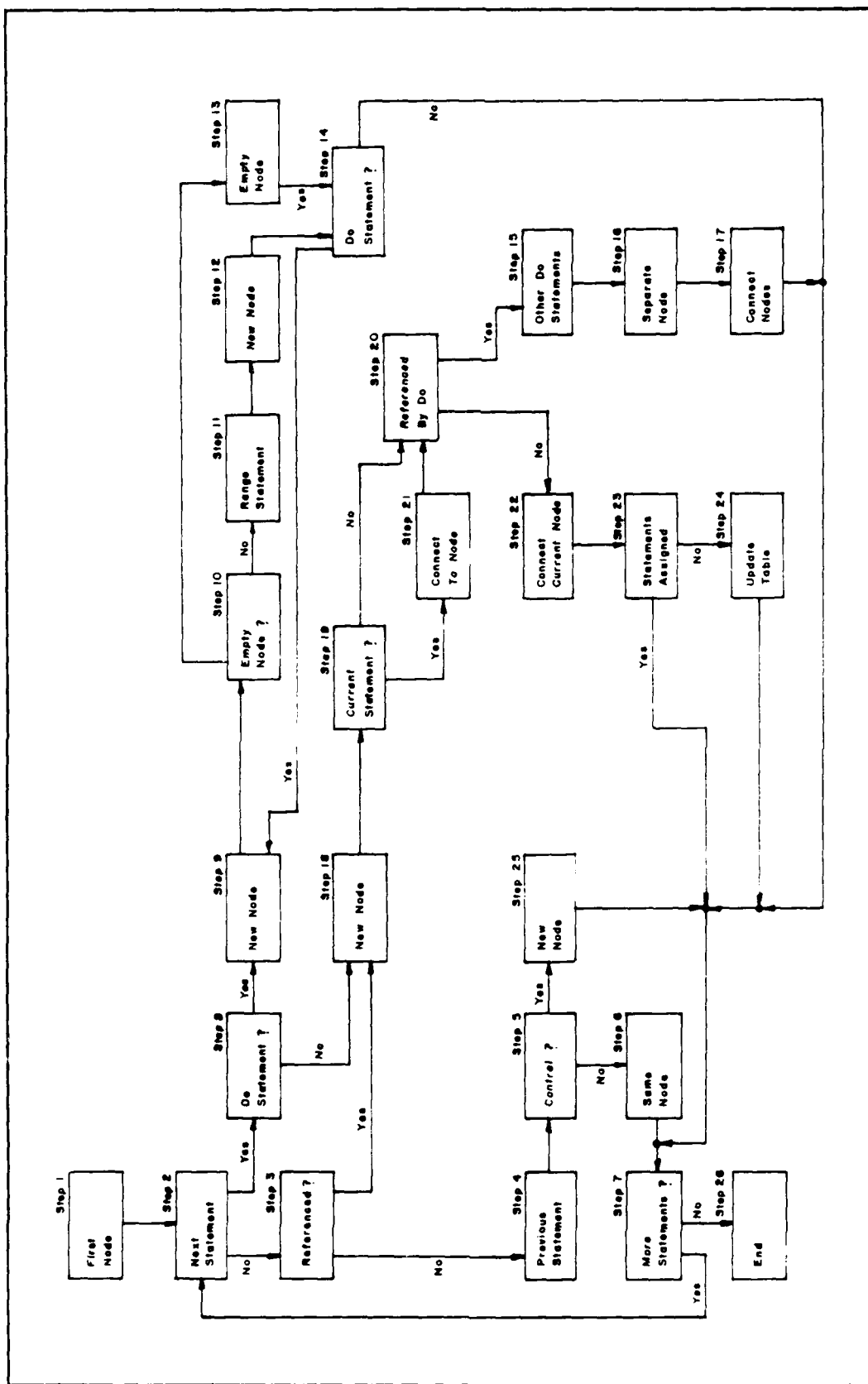


Figure 2-2: Flowchart For Constructing a Program Graph  
[Ref 12: 491]

Type 2 statements are the destination of one or more control statements. They automatically become a separate node. If the control node that references the type 2 statement has already been assigned to a node, then the connecting link can be made. However, the information must still be stored in the event that another control statement, that has not yet been encountered, also refers to that statement. If no control statements have been encountered that refer to the type 2 statement in question, then that information is also stored so that the link can be made later. Type 2 statements also form simple nodes.[Ref 12:490]

Type 3 statements are also assigned to a separate node. The node cross reference table is updated to reflect any links that can now be made with statements previously assigned to nodes and also with references to statements not yet assigned to any nodes. Note that a type 3 statement can also be a type 2 statement because it can be the destination of another control statement. Since all type 3 statements automatically become separate nodes anyway, this problem is solved by merely updating the table. Type 3 statements always form complex nodes.[Ref 12:490]

Type 4 statements are handled similar to the other control statements by assigning them to a separate node. The difference is that the nodes containing type 4 statements are simple nodes instead of complex nodes. The statement referenced by the DO statement always follows the DO statement itself so the end of the loop isn't quite as difficult to track. However, it is still necessary to determine if it is also a type 2 statement. Here again, as in type 3 statements, it does not represent a special problem since it is already a separate node.[Ref 12:490-492]

Type 5 statements mark the end of the DO loop and as such are formed into a separate complex node. However, if this statement is the only statement in the range of the do-loop then a separate empty or null node is set up between the nodes formed by the type 4 and the type 5 statements.[Ref 12:490-492]

The algorithm functions by analyzing the executable statements of a user's program. Any comments or data declarations are not analyzed. The flowchart consists of 26 steps, each of which are detailed below.

Step 1 forms the first node from the first executable statement.

Step 2 gets the next executable statement and determines if it is a control statement (type 3 or 4). If the next statement is a control statement, the algorithm goes to step 8. If it is not a control statement, the algorithm goes to step 3.

Step 3 determines if the current statement has been referenced by another statement. If it has not, then it is a type 1 statement and the algorithm goes to Step 4. If it has been referenced previously, then the algorithm goes to Step 18.

Step 4 retrieves the previous statement from the program.

Step 5 analyzes the previous statement to determine if it was a control statement. If the previous statement was a control statement, then the algorithm goes to step 25. If it is not, then the algorithm goes to step 6.

Step 6 includes the current statement into the same node as the previous statement and then passes control to step 7.

Step 7 determines if there are any more statements to be processed. If there are more statements, the algorithm goes back to step 2. If there are no more statements, then the next step is step 26.

Step 8 is reached from step 2 if the current statement is a control statement. Step 8 looks at the current statement to determine if it is a DO statement. If it is a DO statement, the algorithm goes to step 9. If it is not a DO statement, then the algorithm goes to step 18.

Step 9 is reached either from step 8 above or from step 14. Step 9 puts the current statement into a new node and then passes the algorithm to step 10.

Step 10 checks the following statements to see if the reference statement of the DO loop is the same as the last statement in the DO loop range. If it is the same, then an empty node is needed and the algorithm goes to step 13. If it is not the same, then the algorithm goes to step 11.

Step 11 scans the following statements to find the range statement and then passes control to step 12.

Step 12 assigns the range statement found in step 11 to a separate new node. The algorithm then goes to step 14.

Step 13 is reached from step 10 when an empty range node is needed. This step creates the empty node and then goes to step 14.

Step 14 is reached from either step 12 or step 13. Step 14 looks at the next statement to see if it is also a DO statement. If it is, then the next step is step 9. If it is not, then the next step is step 7.

Step 15 is reached from step 20 if the current statement is referenced by a DO statement. Step 15 finds the number of other DO statements that reference the current statement and passes the algorithm to step 16.

Step 16 creates a separate node for each additional DO statement so that each DO loop has a distinct beginning node and ending node. The next step is step 17.

Step 17 connects each of the nodes created in step 16 to the node of its respective DO statement. Step 17 then passes control to step 7.

Step 18 is reached either from step 8 if the current statement is a control statement but not a DO statement (type 3), or from step 3 if the current statement is not a control statement but has been referenced by another statement (type 2). Step 18 creates a new node for the current statement and then passes control to step 19.

Step 19 checks the node cross-reference table to see if the current statement is there. If it is not, then the next step is step 20. If the current statement is in the table, then the next step is step 21.

Step 20 is reached from step 19 or from step 21. Step 20 checks the current statement to determine if it is referenced by a DO statement. If it is referenced, then the algorithm goes to step 15. If it is not, then the next step is 22.

Step 21 looks up the proper entry in the node cross-reference table and connects the current statement to the node that is referencing it. The next step is 20.

Step 22 connects the current node to all of the other nodes that reference it or to the destinations in the node cross-reference table if the current statement is a control statement. The next step is 23.

Step 23 determines if all of the statements referenced by the current statement have already been assigned to a node. If they have, control goes on to step 7 to get the next statement. If not, then the next step is 24.

Step 24 updates the node cross-reference table by putting one entry in for each unassigned statement. Then the algorithm goes on to step 7.

Step 25 is reached from step 5 to handle the special case of a type 1 statement that is preceded by a control statement. In this case, step 25 sets up a new node for the current statement and then goes on to step 7.

Step 26 is reached from step 7 if all of the statements have been exhausted. Step 26 is simply the end of the algorithm.[Ref 12:491]

The next article of interest is "A Flexible Development System for Microprogrammable Microprocessors." This article describes an expandable system based on bit-slice technology. This system contains a variable number (up to sixteen) of Register and Arithmetic Logic Units (RALUs) and an equal number of Microprogram Control Units (MCUs). In bit-slice technology, one RALU and one MCU can be combined to make one microprocessor. However, in this system, they are not connected together in a dedicated fashion. The inputs and outputs of each of these devices are passed through an interface unit that is controlled by a general purpose host computer.[Ref 1:159-161]

User programs are written in BASIC and compiled in the host computer. The individual operations to be performed by the BASIC program are matched to microprograms that are to be executed in the microprocessors. Each microprogram represents one instruction. The host computer also contains these microprograms in its main memory and feeds them to the MCUs for execution.[Ref 1:162-164]

This system is dynamic in the sense that the host computer selects the RALU and the MCU that are going to execute each microprogram. Since the host computer also contains the microprograms, the instruction sets executed by the microprocessors can vary during execution.[Ref 1:161,164]

Rauscher and Agrawala

"Dynamic Problem-Oriented Redefinition of Computer Architecture Via Microprogramming" discusses a technique for architecture redefinition using customized microprograms. This article establishes execution time and program size as the performance to be optimized in constructing the microprograms. The algorithms presented by the article define procedures for automatically doing the following:

- a) analyzing, at compile time, the intermediate language representation of a program to determine which sections can be made into primitives and represented by a single "machine language" instruction.
- b) generating, at compile time, the microinstructions to interpret these "machine language" instructions.

These algorithms take advantage of the fact that, even for large programs,

- a) instructions generally fall into certain sequences of operations and
- b) small parts of a program account for most of its execution time.[Ref 14:1007-1008]

Each object program is provided with its own set of microcode that is loaded into the control store of the computer just prior to execution. This technique has an obvious shortcoming in a multiprogramming environment as the microcode must be changed at each context switch. However, for programs that consume large amounts of processor time, the use of the processor itself can be greatly optimized.[Ref 14:1007]

#### Fuchs and Johnson

The article "An Expandable Multiprocessor Architecture for Video Graphics" proposes a computer architecture that is optimized for computer processing of video images. The computer system described is composed of a central controller and numerous individual processing

units. Each processing unit does all of the processing for a small subset of the total picture area.[Ref 4:64]

This architecture is not dynamically alterable, nor do the processors have a variable word length. However, it does illustrate an application where numerous processors are executing independent calculations for a single application with a time coordinated solution. It also is an architecture that is optimized for the types and quantities of calculations involved in computer image processing.[Ref 4:58-59]

### III. Computer Generated Imagery Software

#### Computer Generated Images

Computer image generation is the process of taking digitized descriptions of objects and creating a visual scene in the proper perspective for display on a CRT screen. The objects to be displayed are terrain features (mountains, valleys), static objects (buildings, bridges), and moving objects (airplanes). Every three dimensional object to be displayed is divided into a finite number of flat surfaces. This means that round objects must be approximated by dividing the curved surface into some number of flat surfaces. Each flat surface is then defined by identifying the endpoints of the lines that describe or define each edge. These lines are called edges and the flat surfaces are known as faces. Any regular rectangular object, such as a building, would be made up of six faces (bottom, top and four sides).

If no other information except the definition of the edges were given, the building would appear hollow and transparent. In other words, the inside walls would be

visible from outside the building. When color information is added to the faces, a dilemma occurs. The walls are now opaque and there is confusion as to which surface is visible. This contradiction is resolved by assigning a priority to each face. The priority is assigned based on the position of the surface with respect to the viewer. A face that is obscured by another face is given a lower priority than the face that is closer to the viewer. Any portion of a high priority surface that lies between the viewpoint and a low priority surface will mask out that portion of the low priority surface.

It is the job of the computer image generation system to take the description of all items in the data base and determine which edges are in the field of view (FOV). The system then calculates the perspective of each visible edge from the viewpoint. This edge is then projected onto the viewing window. If any of the edges extend outside the viewing window, they are clipped off to the edge of the viewing window. Sophisticated image generation systems also perform gradual shading on the flat sides of the curved surfaces to present a more realistic looking curved surface. Other possible enhancements include sun angle effects such as shadows and the application of texture to selected surfaces.

The viewing window is divided up into raster lines and the raster lines are divided up into picture elements. The final step in the image generation process is to convert each surface to its corresponding picture elements and then to define a color for each element.

### Scene Generation Software

The scene generation software used in this study is a part of a much larger system of software called the Data Base Development System (DBDS). The flow diagram for the system is shown in Figure 3-1. The purpose of the DBDS is to create, modify and test CGI data bases in a non-real-time manner. It provides the full range of data base creation capabilities allowing all but the final verification to take place independent of the video hardware. Processing can take place interactively, allowing the user to vary parameters and view the results as the data base is being created. Once the data base is tested and verified, it is used in a real-time system for generating dynamic visual scenes.



The processing done by the scene generation software in the DBDS is done by hardware in the real-time system. The scene generator software in the DBDS simulates the actual hardware in the real-time system.

The scene generation system is comprised of six different parts, each with a specific function. The functions of each processor is as follows:

#### Face Compression

Compresses the data describing each face into a format that is more efficient for the remaining calculations

#### Controller

1. Reads input data (Visual Parameter File)
2. Reads data base files (Environment File, Priority Data File)
3. Reads Fixed data files (color table, light parameter table, texture map)

Frame 1

1. Computes the direction cosine matrices required to relate earth, viewpoint and model coordinate systems. Figure 3-2 illustrates the coordinate set definition
2. Computes the constants for the view window (to offset the viewpoint matrices to upper left-hand window corner)
3. Computes relative position of the viewpoint to the region.
4. Tests each data base region to see if it lies in the field of view (FOV)
5. Tests each region in the field of view (FOV) to define the level of detail
6. Computes for static data
  - a) relative viewpoint to subregion centroid
  - b) sun vector relative to the subregion
  - c) matrix for rotating static subregion data to the view window
7. Computes for moving models
  - a) moving model origin
  - b) moving model direction cosine matrix

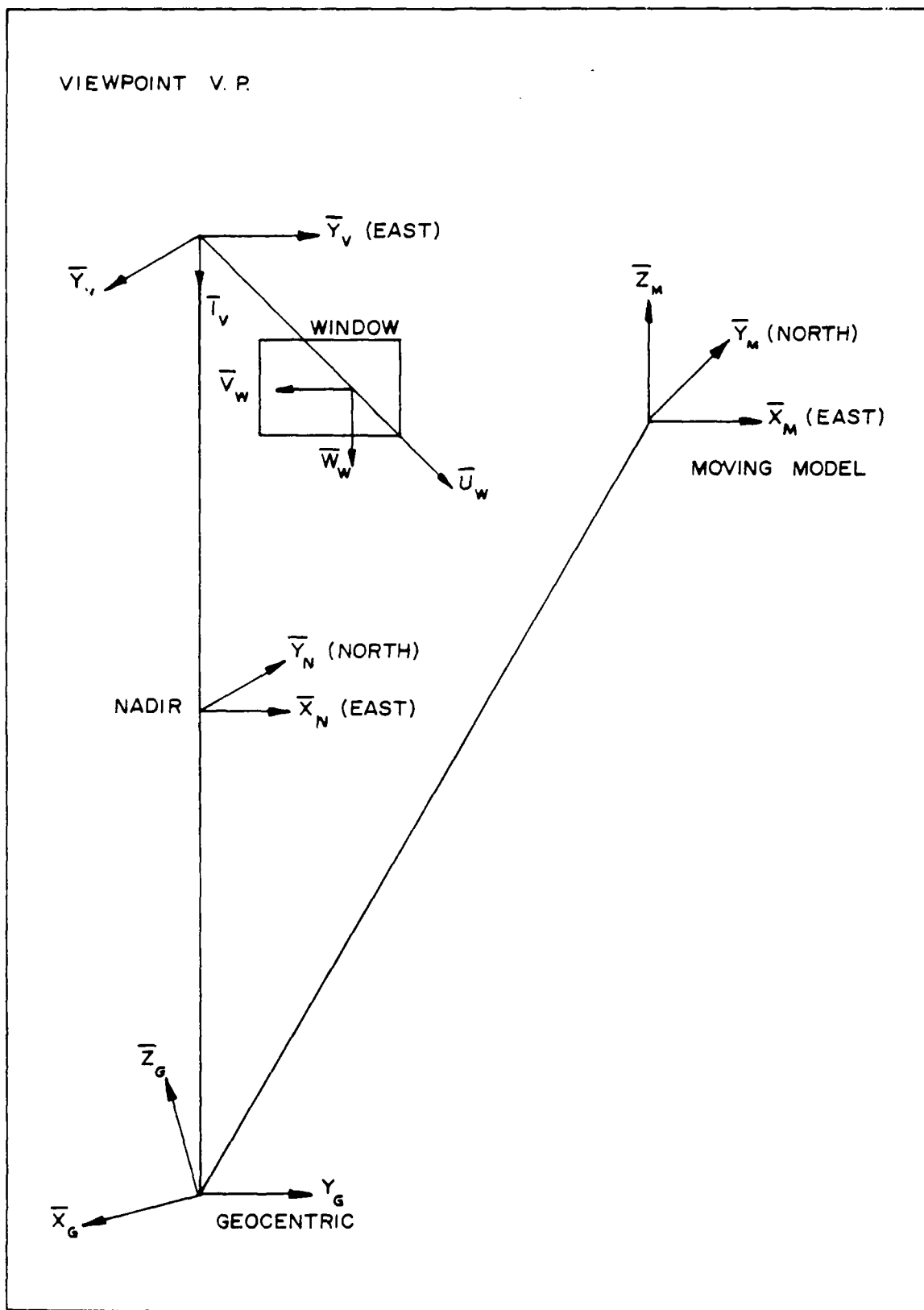


Figure 3-2: Coordinate Set Definition

c) matrix to rotate moving model data to the view window

d) sun vector rotated to the moving model coordinate system

#### Frame 2

1. Reduces the scene data to raster line and line element on the display window
2. Lists the active faces (for use in determining visual priority from the view point)
3. Calculates color and texture data for use in Frame 3

#### Priority Processor

Generates a list of the priority numbers for each face in the active face list.

#### Frame 3

1. Selects the best edges for each element in the line
2. For each element in the line, calculates texture shading coefficients
3. Calculates color intensities for each element in the line
4. Outputs the line to the video hardware

Appendix A contains a summary of all of the modules arranged alphabetically. For each module, there is a list of all of the modules that it calls and a list of all of the modules that it is called by. In addition, there is an identifier that denotes which of the six major programs contains or calls that particular module. The modules are divided into application modules and system modules. The application modules are those written specifically for a particular application. The system modules are a part of the operating system library and their source code is not available. The system modules are not a part of this analysis other than the reference to them in this appendix.

Appendix B contains a list of all of the application modules with a short description of the function of each.

Each of the six programs in the scene generator software executes independently. The disk files are used for intertask communication and the COMMON areas are used for intratask communication. The first program, called FRAME1, reads its input data from a disk file into COMMON and then performs its functions. When FRAME1 finishes, it writes all of its output data back out to a disk file to be read in by the next process. This procedure is followed by

all of the subsequent processes until at last a scene is output to the video hardware.

### Required Data

There are two aspects of the software that are important in this analysis. The first is the characteristics of the variables and data being used in the programs. The second is the characteristics of the code itself.

Characteristics of the Data. For purposes of the following discussion, the term "variable" applies to all memory positions used during execution that do not contain instructions. All of the available information about the variables used by the scene generator software can be found from the storage map at the end of the compiled listings. This information includes the variable type (real, integer or logical) and how much storage is allocated to it (halfword, fullword, double). This measure of length is rougher than what is ideally desired for this analysis because it only gives the size allotted by the compiler. This measure is usually given in halfwords (2 bytes) and

multiples of halfwords and is assigned by the compiler based on information supplied by the programmer. The compiler makes no attempt to minimize the amount of memory required for each variable.

For example, if the FORTRAN compiler allows one full word (4 bytes) for the storage of a variable, that means there are 32 bits available. However, if that variable only assumes values of 1 and 2 during execution then there are many bits that are unused. These unused bits are exactly what is to be avoided by using a variable architecture.

Additional information on the range of values required for each variable could be obtained by printing the value of each variable every time it is changed during execution. This process would require recompiling each module with additional output statements that directed the values of interest to an off-line file. However, recompilation of all modules is not feasible at this time because of the quantity of additional software which would have to be running also. Additional software would be required in order to support all of those functions that lie outside of the scene generator block in Figure 3-1. It is outside the scope of this effort to convert and debug all of this software.

Program Analysis. Each line of code in the software can be classified as either an executable statement or a non-executable statement. The non-executable statements are those which do not generate any machine instructions. There are three general types of non-executable statements.

The first type are comment statements which are completely ignored by the compiler. The second type are used by the compiler to structure and allocate memory. Examples of this type include DATA, COMMON and DIMENSION statements. The third type generate data which requires some memory. An example of this type of statement is a FORMAT statement.

The executing lines of code consist of data manipulation and control type statements. Data manipulation statements include all lines with an arithmetic operation, or which assign a value to a variable, or which perform a test on data. Examples of arithmetic operations are addition (+), subtraction (-), and multiplication (\*). A value assignment is made with an equal sign (=). Examples of tests which can be performed on data include Less Than (.LT.) and Greater than or Equal to (.GE.).

Control statements direct the flow of program execution between alternate paths. Control statements can involve decisions as in the case of IF or DO, or they can be unconditional like GOTO or RETURN.

The data required from the analysis of the scene generator software includes the type and quantities of the instructions being executed. This information is used to establish the instruction set of the computer being designed.

### Software Analysis

The method used to analyze the code to collect this data is described below.

#### Task 1: Compile all programs and subroutines.

Compiling each module produces a variable storage map. This map shows the name, size and type of all variables used in that module. This information is used to describe the characteristics of the data that must be handled by the dynamic computer system being designed.

Compiling the modules was not a straightforward task. The modules were originally written to run on a Systems Engineering Laboratory (SEL) computer. Later, they were converted to run on an Interdata computer. Now the source code is once again on a SEL computer. The process of getting them to compile on the current computer system required numerous changes to the source code.

When the programs were written to run on the original SEL computer system, the COMMON data areas were put into INCLUDE files that were combined with the rest of the source code at compile time. However, the Interdata compiler did not support the INCLUDE function. Consequently, when the programs were converted to run on the Interdata computer, a special preprocessor was written to perform the include function. This processor created a temporary intermediate file from the source code and a second file that contained all of the INCLUDE files. This intermediate file was then processed by the regular FORTRAN compiler.

Now that the programs are to be compiled on another SEL computer, it was necessary to write a preprocessor to perform the same function. It was not possible to simply put the common definitions back into INCLUDE files since

some INCLUDE files now had the same names as some of the modules.

The syntax of some of the Interdata FORTRAN statements differs from the syntax of the same statements in the SEL FORTRAN. These statements had to be changed in order to get the modules to compile. These changes were in the following three areas:

1. ENDDO. Both Interdata and SEL FORTRAN support the same kinds of DO loops. However, the ENDDO or DO loop termination is different. SEL DO loops all end with an ENDDO or CONTINUE regardless of the type. Interdata FORTRAN DO loops end with a different ENDDO based on the type of DO loop involved. For example, DO FOR ends in ENDDO FOR.

2. IF. Interdata IF statements do not require the use of the word THEN following the IF clause. SEL FORTRAN requires the word THEN for proper syntax.

3. Hex Data Declarations. Interdata FORTRAN supports three types of hex data declarations, X, Y, and Z depending on the conditions under which the data is used. SEL FORTRAN only supports the X type.

Task 2: Compile a list of all variables and their attributes. The variables used by each module come from four sources. The first source is the local variables that are used only within that module. The second source is the common variables that are shared by other modules and are identified by placing them in a separate area accessible by all of the modules that need them. The third type are temporary variables that are generated by the compiler to hold intermediate results. The fourth source are the constants required for calculations. Only the first two types are of interest here.

The list of variables is constructed with the variable name, the location of the variable, the variable type, the variable size, the variable dimension and the total memory required for storage of all positions of the variable. There may be more than one variable in the list with the same name because different modules can have local variables with the same name with no conflict. Modules can even have local variables with the same name as variables in common as long as that common is not contained in the subject module. The only requirement is that there are not two variables in the same module with the same name.

The location of the variables is given in the list in order to distinguish between variables of the same name. If the variable is in common, the name of that common is given as the location. If the variable is a local variable, the name of the module is given as the location.

The variable type can be integer, real, character or logical. Integer variables in the table are denoted by the letter I in the type column. Likewise, real variables are denoted by R, character by C and logical by L. The variable size is given in increments of bytes. The type and size combinations encountered are as shown in Table 3-1.

Table 3-1: Valid Variable Type and Size Combinations

<u>Type</u>	<u>Size</u>
Integer	2 bytes
Integer	4 bytes
Real	4 bytes
Real	8 bytes
Logical	4 bytes
Character	1 byte

Each variable is identified in the variable storage map at the end of each module's listing as being either a variable or an array. Arrays are identified in this list by the word ARRAY in the usage column. The dimension of each variable is determined from the definition of the variable

in the source code. Two dimensional arrays are converted to a single dimension by multiplying the two dimension values. The total memory required for the storage of each variable is calculated by multiplying the dimension by the size.

Six lists are constructed, one for each of the six main parts of the scene generator software. Each list is compiled by extracting the pertinent information from the variable storage map at the end of each listing. The list for each part of the scene generator software is started with the listing for the main program of that part. Each list is then completed with the variable storage map of all subordinate modules of that part of the scene generator. Each variable in each variable storage map is compared to the variables in the appropriate list. Variables from the subordinate modules are added to that list if:

1. they have a different name from all the other variables that are already in the list or
2. they have the same name as a variable that is already in the list but are in a different location.

The complete tables are given in Appendix C. The data is summarized in Table 3-2 and Table 3-3.

Table 3-2: Summary of Variable Data

Program Name	Variable		Variables	Totals		Percent	
	Type	Size		Dimension	Total	Variables	Storage
FACCOM	Integer	2	2	14,096	28,192	3.45	57.56
	Integer	4	53	5,191	20,764	91.38	42.40
	Real	4	1	1	4	1.72	0.01
	Real	8	2	2	16	3.45	0.03
	TOTAL		58	19,290	48,976	100.00	100.00
FRAME1	Integer	2	13	1,838	3,676	2.77	10.31
	Integer	4	180	3,838	15,352	38.30	43.07
	Logical	4	16	47	188	3.40	0.53
	Real	4	248	4,071	16,284	52.77	45.69
	Real	8	13	18	144	2.77	0.40
	TOTAL		470	9,812	35,644	100.01	100.00
FRAME2	Integer	2	25	4,128	8,256	3.65	6.38
	Integer	4	335	18,691	74,764	48.98	57.79
	Logical	4	38	69	276	5.56	0.21
	Real	4	283	11,500	46,000	41.37	35.56
	Real	8	3	10	80	0.44	0.06
	TOTAL		684	34,398	129,376	100.00	100.00
FRAME3	Character	1	8	8	8	0.48	0.00
	Integer	2	435	54,872	109,744	26.33	41.65
	Integer	4	807	24,318	97,272	48.85	36.92
	Logical	4	14	76	304	0.85	0.12
	Real	4	373	14,002	56,008	22.58	21.26
	Real	8	15	19	152	0.91	0.06
	TOTAL		1,652	93,295	263,488	100.00	100.01
SCGEN	Integer	2	10	1,963	3,926	4.81	6.95
	Integer	4	129	6,263	25,052	62.02	44.34
	Logical	4	12	43	172	5.77	0.30
	Real	4	55	6,821	27,284	26.44	48.29
	Real	8	2	8	64	0.96	0.11
	TOTAL		208	15,098	56,498	100.00	99.99
PRIPRO	Integer	2	65	60,757	121,514	20.31	77.28
	Integer	4	238	7,824	31,296	74.38	19.90
	Logical	4	1	32	128	0.31	0.08
	Real	4	11	1,062	4,248	3.44	2.70
	Real	8	5	7	56	1.56	0.04
	TOTAL		320	69,682	157,242	100.00	100.00

Table 3-3: Total Variable Data

<u>Variable</u>		Variables	<u>Totals</u>	Total	<u>Percent</u>	
Type	Size		Dimension		Variables	Storage
Character	1	8	8	8	0.24	0.00
Integer	2	550	137,654	275,308	16.21	39.83
Integer	4	1,742	66,125	264,500	51.36	38.27
Logical	4	81	267	1,068	2.39	0.15
Real	4	971	37,457	149,828	28.63	21.68
Real	8	40	64	512	1.18	0.07
TOTAL		3,392	241,575	691,224	100.01	100.00

The data in Table 3-2 is arranged by major program. The first column of the table is the major program name. The second column is the variable type and the variable size in bytes. The next set of columns gives totals for the number of distinct variable names, the sum of all the dimensions of those variables, and the total memory required for storage of those variables.

The last set of columns shows two different pieces of information concerning the relative occurrence of the separate types of variables. The first of these columns shows the percent of occurrences of that type of variable to the total number of distinct variables. It is calculated as the ratio of each value in the Variables column to the total of that column. The second column of this set shows the

percent of memory that type of variable occupies in relation to the total memory for that module.

It is important to note that there is a subtle difference in meaning between the total number of distinct variable names and the total of their dimensions. The first line of Table 3-2 illustrates this point. The program FACCOM contains only two variables that are two byte integers. However, their total dimension is 14,096. Instructions in FACCOM only have to deal with two variables but the total memory requirement for them is over fifty percent of the total memory of that program. This means that the handling of the dimension function, or indexing, is the significant factor in the processing required to manipulate these variables.

Task 3: Itemize instructions by function and variable size. The individual lines of code of each module are analyzed to determine what functions are being performed. This data is used to determine what types of machine instructions would be required in order to perform those functions.

Each operation in an executable statement is categorized according to its function. If data manipulation is involved, then the operations are further broken out according to the size and type of the data involved. The result is a count of the total number of individual operations (integer additions, subtractions, etc.) performed within that module.

A large number of the source statements contained operations involving more than one size or type of variable (mixed mode operations). In order to make the data consistent across each operation, the standard rules for FORTRAN parsing were used. That is:

- 1.) all statements are evaluated from left to right
- 2.) multiplication and division take precedence over addition and subtraction
- 3.) in operations involving two different types of variables (integer, real, etc.) each variable is first converted to the higher order type.

In addition, other assumptions were made in order to simplify the data collection and the resulting design:

1. Dimensioned variables were treated like regular variables. That is, unless an arithmetic operation occurred within the index, no special operation was counted. The assumption is that the instructions being designed into the dynamic architecture computer would have the capability to handle variable indexing without any additional overhead. If an arithmetic operation was performed within the index, then that arithmetic operation was counted. For instance, the variable MAP(2,I-7) contains a subtraction operation on a four byte integer variable that would be counted as a separate operation. However, the indexing of the variable MAP would not be counted as an operation.

2. Assignment operations are considered to be of the same size and type as the size and type of the variable on the left side of the assignment sign.

The complete data tables are in Appendix D. The data is summarized in Table 3-4, Table 3-5, and Table 3-6.

Table 3-4 contains the total number of occurrences for each operation for each type of variable. The first column contains the symbol or description of the operation being tabulated. The second column contains the number of

Table 3-4: Itemization of Variable Operations Data

Operation	I*2	%	I*4	%	L	%	R*4	%	R*8	%
+	206	5.19	510	11.21	0	0.00	232	7.85	11	10.19
-	46	1.16	160	3.52	0	0.00	253	8.56	25	23.15
*	39	.98	71	1.56	0	0.00	329	11.14	19	17.59
/	3	0.08	39	0.86	0	0.00	101	3.42	18	16.67
**	0	0.00	2	0.04	0	0.00	30	1.02	8	7.41
=	1599	40.29	2327	51.15	71	13.76	1312	44.41	19	17.59
Arith IF	4	0.10	3	0.07	0	0.00	0	0.00	0	0.00
Logic IF	770	19.40	533	11.72	282	54.65	251	8.50	2	1.85
ELSEIF	59	1.49	34	0.75	4	0.78	13	0.44	2	1.85
.EQ.	622	15.67	322	7.08	0	0.00	38	1.29	0	0.00
.NE.	325	8.19	124	2.73	0	0.00	15	0.51	0	0.00
.GT.	63	1.59	172	3.78	0	0.00	91	3.08	0	0.00
.LT.	23	0.58	52	1.14	0	0.00	124	4.20	0	0.00
.GE.	20	0.50	26	0.57	0	0.00	56	1.90	0	0.00
.LE.	8	0.20	33	0.73	0	0.00	18	0.61	4	3.70
.AND.	123	3.10	72	1.58	9	1.74	73	2.47	0	0.00
.OR.	59	1.49	69	1.52	139	26.94	18	0.61	0	0.00
.NOT.	0	0.00	0	0.00	11	2.13	0	0.00	0	0.00
Totals	3969	100.01	4549	100.01	516	100.01	2954	100.00	108	100.00

Table 3-5: Summary of Variable Operations Data

Variable Type	Total Operations	PerCent
I * 2	3969	32.81
I * 4	4549	37.61
L	516	4.27
R * 4	2954	24.42
R * 8	108	0.89
Total	12096	100.00

Table 3-6: Summary of Other Operations Data

<u>Operation</u>	<u>Quantity</u>
GO TO	1260
GO TO ASSIGN	4
Computed GO TO	1
DO n	302
DO FOR	150
DO FOREVER	1
DO UNTIL	20
DO WHILE	12
LEAVE	2
Procedure Call	716
Subroutine CALL	703
READ	4
WRITE	484
FORMAT	372
SELECT CASE	5
CASE	18
ASSIGN	47
REWIND	8
RETURN	242
STOP	3
	----
Total	4354

occurrences of each operation that uses two-byte integer variables. The third column gives the percent of occurrences of that operation on two-byte integer variables. This percent figure is calculated based on the total number of two-byte integer operations.

The remaining columns in the table are paired just like column two and column three. Each remaining pair of columns contains the data for four byte integer, logical, four byte real and eight byte real variables respectively.

Table 3-5 contains a summary of the total operations by variable type. It also contains the percent of operations by variable type relative to the total number of operations.

Table 3-6 gives the total number of non-data operations that are contained in the scene generator software. These operations are necessary to the functioning of the software but they do not manipulate any data other than counters or internal variables.

#### IV. Dynamic Architecture Computer Design

A suitable dynamic architecture computer design could be developed based strictly on the storage requirements of the various types of data used throughout the software. In this case, the scene generator software is being used as an example. An analysis of the storage percent column in Table 3-2 suggests assigning a priority to each variable type based on the percentage of its memory requirements. In approximate terms, this means that the two-byte integer variables require about the same amount of memory as the four-byte integer variables. It also means that the four-byte real variables require only about half of the memory space as both of the integer cases. The memory requirements of the other types of variables are all much smaller in comparison. This approach might yield a set of dynamic computer configurations with the following processors:

- 1.) eight byte wide real
- 2.) two 4-byte wide integer
- 3.) two 2-byte wide integer + one 4-byte wide real

A fourth possibility would include logical operations with:

4.) one 4-byte wide integer + one 2-byte wide integer  
+ one 2-byte wide logical

All of the logical variables declared within the modules are four bytes wide. However, in every instance that they are used, one byte would suffice. Therefore, all of the logical operations could be handled by the two-byte wide integer configuration.

A very small number of character variables exist within the entire system of programs. However, their existence is misleading because there are no character manipulations in the software at all. The character variables in this application are only declared and placed in common for diagnostic and future expansion purposes. Therefore, it is not necessary to consider them in this design.

Even if there had been some instructions that used character variables, the total number of character variables would still be too small to warrant a configuration with a separate processor for character manipulations. Only if

there were a large number of character instructions would a separate processor be practical.

An eight-byte wide processor for real variables is desired in the computer design because of the ease of implementing eight-byte wide floating point instructions in an eight-byte processor. That is, without this wide processor, there would be a significant increase in the number and complexity of individual operations necessary to process the eight-byte wide data. The total number of operations would increase because the complete width of the data would not fit in the processor all at once. The complexity of the operations would increase because it would be necessary to keep track of all of the carry-ins and carry-outs between the operations. However, because of the relatively low number of eight-byte real variables, it would be expected that the computer would spend very little time in this configuration.

The two four-byte wide integer processors in the second configuration are desirable because of the high percentage of four-byte integer variables.

The third configuration is also desirable because it can process twice as many two-byte integer variables as it can four-byte real variables. Since there are twice as many two-byte integer variables as there are four-byte real variables, the third configuration provides a balance between these two variable types. The second and third configurations are also balanced between themselves.

The above design is based on the total storage requirements of all of the types and sizes of variables. It does not account for the fact that the relative proportion of the types of variables is different when based on the number of distinct variable names. When the data is analyzed from the point of view of different variables instead of just storage requirements, it is observed that there is a bigger difference in the number of two-byte integer and four-byte integer variables. In fact the number of four-byte integer variables is three times the number of two-byte integer variables. The number of four-byte real variables is still roughly half the number of four-byte integer variables but it is now about twice as many as the two-byte integer variables. This suggests that there might be some benefit in expanding the total width of the dynamic computer architecture to accommodate more four-byte integer processors.

This could be easily accomplished by simply adding one four-byte integer processor to each of the proposed configurations. The result would be as follows:

- 1.) one 8-byte real + one 4-byte integer
- 2.) three 4-byte integer
- 3.) two 2-byte integer + one 4-byte real + one 4-byte integer
- 4.) two 4-byte integer + one 2-byte integer + one 2-byte logical

This design would place at least one four-byte integer processor in every configuration. Since more than half of the variables in the software are four-byte integers, this architecture would permit at least one four-byte integer process to be executing whenever any other process was executing. This would be desirable if the data regarding the number of variables is representative of the type of processing that is required.

So far the design has been based on the types and quantities of data encountered. The number and type of data manipulations (instructions) is not in the same ratio.

Table 4-1 shows a comparison of the percent of occurrences of each type of variable, the percent of memory required for each type of variable, and the percent of occurrence of instructions for each type of variable. This data is repeated from previous tables in Chapter 3. In addition, a relative ranking is given for each percent. This number is merely the rank order of each percent within that column.

Table 4-1: Relative Occurrences by Variable Type

<u>Variable type</u>	<u>Variables</u>		<u>Storage</u>		<u>Instructions</u>	
I * 2	16.21	3	39.83	1	32.81	2
I * 4	51.36	1	38.27	2	37.61	1
L	2.39	4	.15	4	4.27	4
R * 4	28.63	2	21.68	3	24.42	3
R * 8	1.18	5	.07	5	.89	5

Table 4-1 itemizes the analysis techniques employed in this study and summarizes the data gathered by each technique. It is important to note that the three variable types occurring most often are the same regardless of the method of analysis. That is, the logical variables and the eight-byte real variables always occur at a much smaller rate than the other three variable types. Therefore, the data collected supports design number two above.

The second design has a total of 13 processors divided up as follows:

- seven 4-byte integer processors
- three 2-byte integer processors
- one 2-byte logical processor
- one 4-byte real processor
- one 8-byte real processor

The scene generator software is currently executing in a Systems Engineering Laboratory (SEL) 32/70 computer. In order to achieve a gain in execution speed over the SEL computer, the dynamic computer being designed should have at least equivalent capabilities in areas such as the instruction set. The functional classification and number of instructions of the SEL 32/70 computer system is given in Table 4-2. The variable types and the instructions used to implement various software functions are discussed in the following paragraphs.

Integer variables are called fixed point variables in the SEL computer vendor's literature. The SEL computer systems handle four sizes of integer variables. They are byte (1 byte), halfword (2 bytes), word (4 bytes), and doublewords (8 bytes).

Table 4-2: SEL 32/70 Instruction Repertoire

<u>Classifications</u>	<u>Number</u>
Fixed Point Arithmetic	30
Floating Point Arithmetic	8
Boolean	17
Load/Store	29
Bit Manipulation	8
Zero	5
Shift	13
Interrupt	13
Compare	11
Branch	9
Register Transfer	13
Input/Output	10
Control	16
Hardware Memory Management	4
Writable Control Store	<u>3</u>
Total	189

There are 30 fixed-point arithmetic instructions of which five deal with word operands and five deal with halfword operands. The rest of the fixed point instructions deal with byte operands, doubleword operands, register operands, immediate operands, and the miscellaneous functions called extend sign and round register.

Real variables are referred to as floating point variables by the SEL computer manufacturer. Floating point variables come in two types in this machine. The floating point word variables are four bytes long and the floating point doubleword variables are eight bytes long. The format of the floating point variables is illustrate in Figure 4-1.

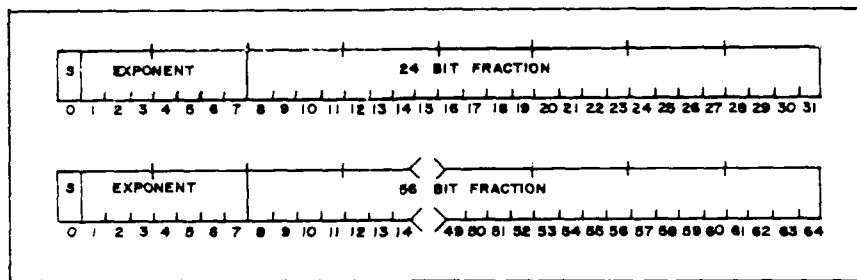


Figure 4-1: Format of Floating Point Variables

There are eight floating point instructions in the SEL 32/70. Four of these are for the four-byte real operands and four are for the eight-byte real operands.

The computer's logical instructions perform the AND function, the OR function, and the exclusive-OR function on bytes, halfwords, words and doublewords. Additional logical instructions are available for performing these functions on registers and masks.

IF statements in the software are implemented in the hardware by performing the necessary arithmetic or logical operations on a temporary work variable and then testing the result for the specified conditions. Instructions are available in the SEL 32/70 computer for doing arithmetic comparisons on all four sizes of variables. Some of the branch instructions have the capability of testing the results of a compare and conditionally branching.

## Dynamic Computer Features

Four different configurations were selected at the beginning of this chapter as being the optimum configurations for the dynamic computer architecture. A graphic example of the four configurations is given in Figure 4-2.

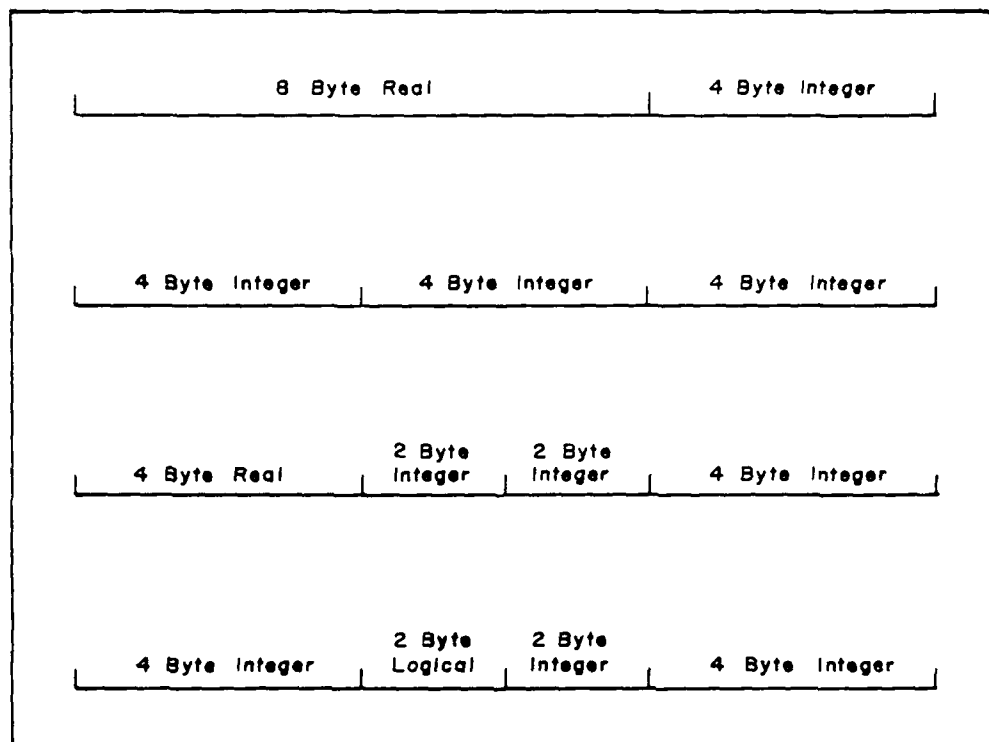


Figure 4-2: Possible Configurations

Format of floating point variables. The format of the floating point variables in the dynamic architecture computer are the same as those of the SEL 32/70 computer. That is, there is one bit for sign, seven bits for the exponent value, and either 24 or 56 bits for the mantissa. Figure 4-1 illustrates the format. This method assures that the variables in the dynamic architecture computer will have sufficient accuracy and storage capacity to achieve adequate performance.

Memory structure. The most important feature of the dynamic architecture computer is the memory structure. There are several methods of implementing an efficient memory access but any approach that is selected must efficiently overcome the problems of changing the word length between configurations.

The changing width of the memory causes a problem in two ways. The first is in the area of memory access. The desired objective of each memory fetch should be to retrieve exactly one variable or instruction regardless of the width of the data being retrieved. If the data were longer than the width of the fetch, then multiple fetches would be needed. If the data were shorter than the width of the

fetch, then only one fetch would be needed but the memory space that was not needed would be wasted space. A memory access mechanism that is able to get the proper number of bits each time it operates is the major benefit to having a variable architecture computer.

The other problem arises when accessing memory to retrieve instructions. If a common instruction format and length are selected so that the instructions for each configuration are the same, then a conflict arises when the working width of the memory changes. If the desired length of the instructions is given to be the length of the shortest variables in memory (16 bits in this case), this length will be found to be insufficient to contain a large enough memory address. If the instruction length is any longer than the shortest variables, then one instruction fetch would require multiple memory accesses.

Obviously there is a conflict between these two requirements. There is the desire to vary the width of memory accesses to make the retrieval of variables efficient. There is also the desire to keep the memory fixed to make the instruction retrieval efficient. Several possible memory structures that provide a partial solution are discussed below.

One possible method would have each memory location the same size as the smallest variable used. This would require multiple memory accesses for most of the variables processed and would not achieve the desired performance.

Another approach would have a separate memory for each size of variable. That is, the 16 bit variables would be stored in a physically different memory than the 32 bit words and the 64 bit words. Whenever the computer architecture is switched from one configuration to another, the memory being accessed would also be switched. This method would make it difficult to have more than one size of variable in the same statement. There would also have to be a mechanism for moving data around between the various memories.

The next method would have a separate memory for data and instructions. Since all instructions would be contained in the same memory, they could all be the same length. However, this approach makes it more difficult to implement any parallel execution paths.

A variation of this method would contain a separate memory for each processing unit in the system. That is, the

64 bit floating point processor would have its own memory and each 32 bit integer processor would have its own memory. This approach would provide for parallel execution paths but there would still be a need for a mechanism for moving data around between the memories.

Another approach would allow different length instructions for each type of processor. This technique would not permit programs and data to be moved around from computer to computer after they were loaded, thus making the system less dynamic. The idea of making the instructions common to all configurations within the computer is called the principle of program universality. The concept of program universality was introduced in the article "Designing LSI Modular Computers and Systems" [Ref 6:9] and was further discussed in the article "Adaptation Properties for Dynamic Architectures" [Ref 13:550].

It is obvious from the above discussion that there is no single solution which will completely satisfy all of the requirements for a dynamic memory structure without having a detrimental effect on some other part of the system. Therefore, the design of the dynamic computer architecture must be based upon a trade-off of the total system

requirements. The basic requirements for the dynamic system may be summarized as follows:

1.) The memory allocated to each variable should be no bigger than what is required to exactly store that variable.

2.) The capability should exist for converting variables from any type to any other type and from any size to any other size.

3.) All instructions should have a common format and length. There are 18 different arithmetic type operations (Reference Table 3-4) and 20 other operations (Reference Table 3-5) identified in the scene generator software. This means that the operation code of the instructions should have at least six bits. Six bits allows a total of 64 distinct operations. It would actually be desirable to allow even more bits in the operation code field for future inclusion of other instructions. For instance, calls to procedures such as Sine and Cosine would improve efficiency if they were implemented as additional instructions.

4.) The memory address portion of each instruction should be sufficient to directly address all variables in

memory. Table 4-3 shows the total number of each size of variable used in the scene generator software along with the number of bits required to directly address that number of variables.

Table 4-3: Summary of Variable Addressing Requirements

<u>Size (Bytes)</u>	<u>Integer Variables</u>	<u>Logical Variables</u>	<u>Real Variables</u>	<u>Total Variables</u>	<u>Bits for Addressing</u>
2	275,308			275,308	19
4	264,500	1,068	149,828	415,396	19
8			512	<u>512</u>	9
				691,216	20

Twenty bits allows a direct memory address of 1,048,576 variables.

5.) The complete memory space must be large enough to physically contain all of the variables and instructions required for the scene generator software. As shown above, 691,216 different memory locations are required for storage of all of the variables. In addition, there are 12,096 variable operations (Reference Table 3-5) and 4354 other operations (Reference Table 3-6). This represents a total of 16,450 separate instructions which must also be placed in memory.

Instruction format. Instructions for the dynamic computer are the same format and size regardless of the configuration. The instruction length was chosen to be 32 bits long. This length allows 9 bits for an operation code, 20 bits for an address, and 3 bits for a memory bank identifier. This instruction format is illustrated in Figure 4-3. The memory bank identifier field allows the computer currently executing that instruction to perform the required data transfer with any of the six data banks.

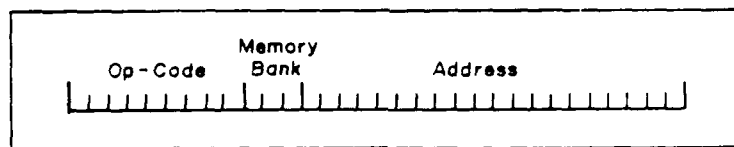


Figure 4-3: Instruction Format

All of the arithmetic and control instructions outlined in the tables in the previous discussion are included. In addition, instructions are provided for changing the architecture from its current state into any of the other configurations.

Look ahead carry. Another feature in the dynamic architecture computer is the implementation of a full

look-ahead carry for all of the types of variables that the system can handle. Without this capability, the computer would have to rely on a ripple carry to perform its arithmetic functions.

Figure 4-4 illustrates how the carry in and carry out function is implemented in the dynamic architecture computer. Multiplexers are provided in order to move the carry out bit to either the carry in portion of the next computing section or to the appropriate condition code register.

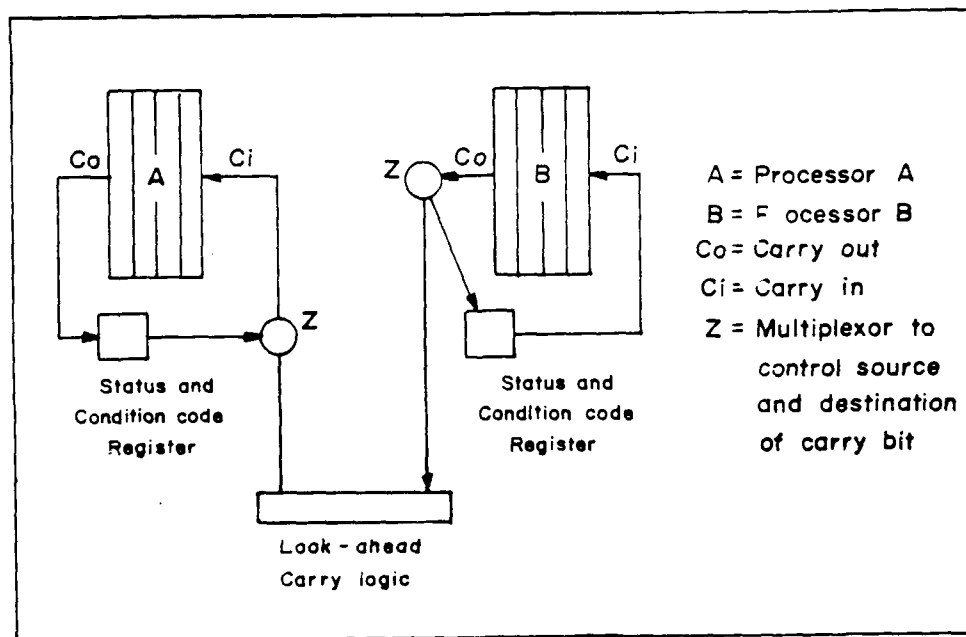


Figure 4-4: Carry-In/Carry-Out Structure

Final design. The final design of the dynamic architecture computer incorporates all of the features described above. Figure 4-5 illustrates a portion of the final design. This figure shows the interconnection of one processing section and the memory banks. The same direct connection exists between all of the processor sections and all of the memory banks. The operation of the dynamic architecture computer is described in the next section.

#### Dynamic Computer Operation

Memory in the dynamic computer is structured into six banks of 16-bit words. The bottom portion of each memory is accessed as individual 16-bit words. The middle portion of each memory is accessed as 32-bit words. That is, each 32-bit memory access retrieves one portion of a 32-bit word from one bank and the other portion of the 32-bit word from the adjacent bank. The top portion of the memory in banks one through four is treated as 64-bit words.

Each memory bank has a Memory Address Register (MAR) and a Memory Data Register (MDR) associated with it. As far

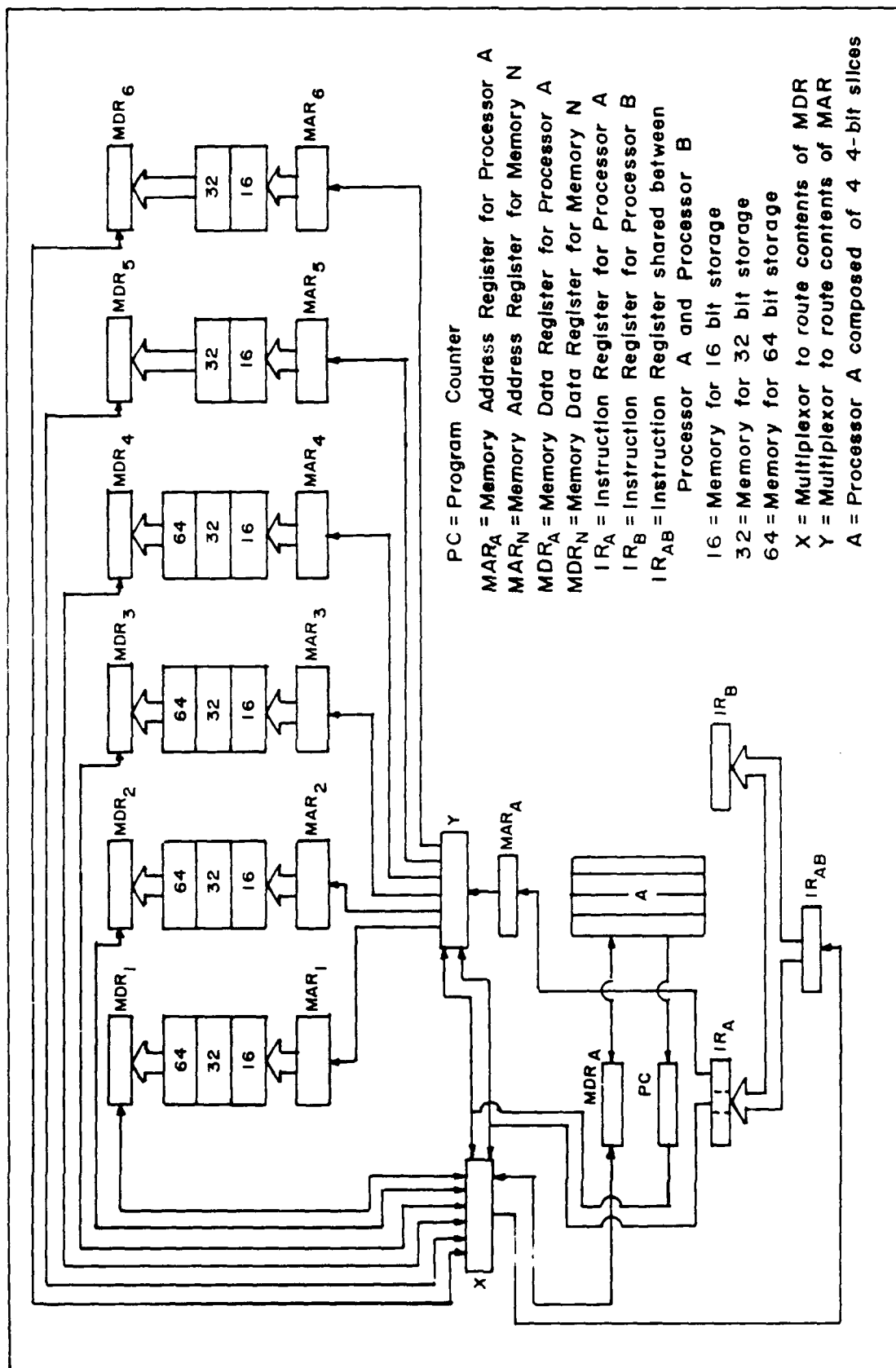


Figure 4-5: Dynamic Architecture Computer

as the individual memory banks are concerned, these registers function in exactly the same manner as the MAR and MDR of an ordinary computer memory unit. That is, each memory bank accesses 16 bits at a time, regardless of the size and configuration of the processor requesting the access. Words that are longer than 16 bits are accessed from the same memory address in adjacent memory banks but not necessarily at the same time. Memory accesses that retrieve different portions of a data word from adjacent memory banks need not occur simultaneously in all of the affected banks. Rather, with this architecture, each memory bank can operate independent of the other banks.

The method for loading and unloading the MAR and MDR as registers differs from an ordinary computer. Instead of having one MAR and one MDR for each processor and memory pair, each memory bank and each processor has a separate set of registers. The MAR of each memory unit is loaded from the MAR of one of the processors or from the Program Counter (PC) of one of the processors. Likewise, the MDR of each memory unit is loaded and unloaded through the MDR of one of the processors or through the Instruction Register (IR) of one of the processors.

Each processing unit is composed of all of the necessary bit slice chips to construct a 16 bit wide processor. Two of these units are combined together to form the 32 bit wide processor when that configuration is called for. Likewise, the hardware from four units are combined to create the 64 bit processor. The connections that determine whether or not a particular unit is a processor by itself or is a part of a larger processor are controlled by individual control units.

Each processor is connected to every memory through a separate data path. This gives each processor the capability to access each memory directly. This direct access capability could have been provided with a data bus which would also facilitate moving data from one memory to another. However, the bus approach would slow down the operational speed of the dynamic computer by forcing all memory transactions to wait for a turn on the bus. The assumption being made with the direct connection approach is that the majority of the variables accessed by a given processor will be contained in a single memory unit. Since there are the same number of memory units as there are processors, each memory may be thought to be most directly associated with a given single processor. If a processor

needs to access a variable in a more distant memory, it can still do so without the time penalty associated with a bussed approach.

Memory access conflicts are handled by having a separate MAR and MDR for each processor as well as each memory. When a particular processor needs to access a particular memory, the multiplexer associated with that processor's MAR first checks to see if the MAR for that memory is empty. An empty MAR signifies that no other processor is using that memory at this time. If the MAR is empty, the processor transfers the memory address contained in its MAR into the MAR of that memory. The memory then proceeds with the transaction by transferring the contents of either the processor's MDR or the contents of the memory's MDR depending on whether a read or a write was requested. If the memory is currently busy with another transaction, this will be signified by a non-empty MAR in the memory. The processor must then wait until the previous transaction is complete before proceeding.

The Program Counter (PC) and the Instruction Register (IR) may also utilize the memory's registers in order to retrieve instructions. The instruction fetch mechanism

follows the same procedure for accessing a memory that the processors follow. Control circuits within the multiplexers remember with which processor or instruction register the memory is currently transacting.

Instruction fetches are implemented by utilizing a 32 bit instruction buffer (IB). Two adjacent processors share a single instruction buffer. This instruction buffer is filled from two adjacent memories whenever an instruction is required by either processor.

Status and condition codes are normally stored in a separate register for each 16-bit processor. However, when two 16-bit processors are configured to act as a single 32-bit processor, the condition codes are passed between the two 16-bit processors and stored in a single condition code register.

The floating point processors are configured such that the exponent portion (the first eight bits) of the floating point variables always occurs in the same eight bits of hardware. When a floating point operation is being performed, these eight bits can be electronically separated from the remaining 24 or 56 bits in order to make the handling of the exponent function easier.

## Summary

The dynamic architecture computer can change its configuration to match the processing requirements of the program currently being executed. The design is optimized to execute a specific software package with a minimum of memory access time.

There is sufficient detail in the design to make some timing estimates and to compare these estimates with the execution requirements of a single, serial processor. This analysis is described in the next chapter.

## V. Analysis and Conclusions

### Analysis

The thrust of this thesis is to present the design of a dynamic architecture computer that will realize a speed improvement over a conventional architecture computer. The design employed has at least two processors available for parallel execution at any time. If the software also has two parallel paths to execute, then it is natural to assume that this computer will now execute the same software in half the time that it would take to execute in a conventional architecture computer. However, there are several factors that must be considered in calculating the dynamic computer execution time.

The first factor is to make an allowance for the switching time for changing the configurations. This point can be resolved by assuming that the time required to switch configurations is at least no greater than the time required to execute any of the other arithmetic or control operations. In the variable architecture computer, the configuration change would take place during one time frame

and then two operations would execute in parallel during the next time frame. In a regular architecture computer, one operation would have been executed during the configuration switch and the other operation would have been executed during next time frame anyway.

The second factor is that it is not necessarily true that there will always be two compatible operations ready to execute. It is possible that the processes that are ready cannot execute in the same configuration given the combination of processors that have been established. In other words, there may not be a four-byte integer operation ready to execute whenever there is an eight-byte real operation executing.

In order to address this problem, it is necessary to know what instructions would occur simultaneously within the code. To do this, it would be necessary to exactly model the program flow for all of the programs in the system in question. Since some of the scene generator programs used as the example are very unstructured, it would be difficult to achieve an accurate program flow graph without extensive rewriting of the code.

Given that an accurate program flow graph is not practical, it is still possible to approximate the speed with which the code might be executed in the variable architecture computer. In order to do this, several more assumptions are required.

First, assume that each identified operation is executed in a single machine instruction. This is a reasonable assumption since the instruction set of the dynamic computer has been designed to provide this capability.

Second, assume that each machine instruction takes exactly the same amount of time. This is a reasonable assumption for all operations except possibly the floating point operations.

Third, it is necessary to assume that the statistical data collected by the analysis discussed above fairly represents the overall operation of the system of programs. Therefore, individual module deviations will average out over the entire system. This also means that the relative percentages of operations of various types will remain the same even though many individual instructions will be executed multiple times.

This final assumption acknowledges the fact that individual modules may have a high number of operations that are of the identical type and size. When this happens, a certain number of those instructions cannot be executed in parallel since not enough processors would exist in any of the configurations. However, modules such as this would still be executable in parallel with another module that has a high number of different type of operation.

Given these assumptions, it is now possible to make the following statements:

1. When a configuration is selected to perform operations on a certain type of variable, the other processors in that configuration will have operations available to perform. For example, when configuration number one is selected in order to execute eight-byte real operations, there will also be enough four-byte integer operations ready to execute so that no processing resources are idle.

2. The order of operations is not significant. This means that enough operations are available for each

configuration, such that the dynamic computer can remain in that configuration for a relatively long time when compared to the configuration switching time. This makes the configuration switching time insignificant.

The best case analysis can now proceed as follows:

1. Select a particular type of variable to operate on
2. Select the configuration that will execute that type of operation
3. Cycle the dynamic computer enough times in that configuration to execute all required operations on that particular type of variable
4. Select a different type of variable and repeat steps 1 through 4.

Table 5-1 summarizes the total number of operations required for each variable type. Table 5-2 summarizes the configurations and the types of operations available in each configuration. Table 5-3 details the results of this analysis performed by selecting first configuration number one, then configuration number two, followed by configuration number three and finally configuration number four.

Table 5-1: Summary of Processors and Operations

Variable Type	Total Operations	Total Processors
I * 2	3969	3
I * 4	4549	7
L	516	1
R * 4	2954	1
R * 8	108	1
TOTAL	12,096	13

Table 5-2: Summary of Configurations and Operations

Configuration	Processors	
	Number	Type
1	1	eight-byte real
	1	four-byte integer
2	3	four-byte integer
3	2	two-byte integer
	1	four-byte integer
	1	four-byte real
4	2	four-byte integer
	1	two-byte integer
	1	two-byte logical

Table 5-3: Detailed Execution Analysis

Configuration	Total Instruction Cycles	I*2	I*4	L	R*4	R*8
1	108	3969	4549	516	2954	108
Balance		0	108	0	0	108
4	516	3969	4441	516	2954	0
Balance		516	1032	516	0	0
3	2954	3453	3409	0	2954	0
Balance		5908	2954	0	2954	0
2	152	0	455	0	0	0
Balance		0	456	0	0	0
Total	3730	0	0	0	0	0

Selecting configuration number one and then cycling the dynamic computer for 108 operations will execute all of the eight-byte real operations. As can be seen from Table 5-3, this will also execute 108 of the four-byte integer operations. This leaves a balance of 4,441 four-byte integer operations and the rest as they were. The balance for all operations is shown on the next line in the table.

The next configuration selected is number four. This configuration is cycled 516 times in order to execute all of the required logical operations. Also at this time, 516 two-byte integer operations and 1032 four-byte integer operations are executed. The balance of the operations are shown on the next line in the table.

Selecting configuration number three next will execute all of the four-byte real operations. Notice that there is a surplus of two-byte integer operations when this configuration is executed 2,954 times.

Finally, configuration number two will execute the remaining 455 four-byte integer operations. Only 152 machine instruction cycles are required since this configuration has three separate processors for this type of operation.

The total number of dynamic computer instruction cycles required is 3730. The total number of instructions required from a conventional computer is 12,096. Figure 5-1 shows that, given ideal conditions, this dynamic architecture computer can execute the same set of operations in only 31 percent of the time that it would take a conventional computer.

$$\frac{3730}{12096} \times 100 \% = 30.8\%$$

Figure 5-1: Percentage of Execution Time Required by Dynamic Computer

The worst case condition would be the situation where each configuration would execute only one instruction before being forced to change to another configuration. In this case, the total execution time would be approximately twice the time required to execute the same software in a conventional computer.

The actual execution efficiency lies somewhere in between. However, a more accurate estimate of execution efficiency would have to be based on a more accurate picture of

what the software actually does while executing. As mentioned earlier, the true execution path can not be known without reorganizing and rewriting some of the example software. Suggestions for how this might be done are discussed in the section on recommended further research.

### Conclusions

The design and analysis presented in the previous sections do not deal with any of the current technology of software engineering and program structure. Although it is possible to expand this architecture to fit more general purpose applications, it would require that the compiler be very intelligent. The compiler would have to be smart enough to break any program down into clusters of equal size variables and then place them into the proper memory bank. A lot of research has been done in finding techniques to do this. Some of these techniques are discussed in the articles in the literature review section.

It may also be found that it is necessary to assign a priority on memory transactions by the various processing units. The data accumulated in analyzing this one

particular application suggests that memory accesses from processors other than the one most directly associated with a particular memory should have priority over memory access by the directly associated processor. Instruction accesses should have priority over both.

A single instruction buffer for two processors will probably not be enough in a general purpose dynamic computer. However, in this application a significant number of the operations are on 32 bit data. Since this means that two 16 bit processors will be combined together to create one 32 bit processor, these two processors are not really sharing the instruction buffer at all. The large number of 32 bit operations in this application is enough to justify having a single instruction buffer.

More research is required on dynamic computer architectures. However, in the specific application addressed by this study it has been shown that a significant increase in processing speed can be achieved by using a dynamic computer.

## Recommendations for Further Research

The design proposed in this thesis for a dynamic architecture computer was based strictly on the statistical occurrence of various types of variables and instructions. As such, it has several limitations which could be eliminated through further research work. For instance, each operation is counted only once even though it may be executed several times in a loop or in multiple entries into the same subroutine. This is how the instructions would be counted if one copy of all of the programs and subroutines were placed in memory at the same time. However, when the program executes, execution will enter several of the subroutines many times. Since the instructions contained in those subroutines are then executed multiple times, those instructions could be given added weight when it comes to calculating the number of instructions per variable type.

One method of solving this problem using the data collected would be to count the total number of instructions that would be executed if all of the subroutine's CALLs were made. This would require multiplying the numbers for each subroutine by the number of times that subroutine was called. For instance, when a subroutine is called twice

from another program, it is the equivalent of passing through that subroutine's code twice.

Another method of analysis would be to substitute the code from each subroutine directly into the place where that subroutine CALL was made. This procedure would require quite a few variable name replacements as each variable in the CALL statement list could have a different name in the body of the subroutine.

At the same time that the subroutine substitutions were being made, other superfluous code could be eliminated. Lines of code like comment lines, printer output and data declarations would not be executed during real-time so they could be eliminated for the analysis.

An analysis of the inner workings of the scene generator software would make it possible to remove much of the disk input and output. As stated in Chapter Two, the communication between major programs is accomplished by passing data in and out of disk files. It can be seen from the module descriptions in Appendix B that a sizable amount of code exists for the sole purpose of compacting and expanding the data in the disk files. If all

of the main programs were resident in memory at the same time during real-time execution, then most of the disk activity could be eliminated.

The end result of this method would be a continuous stream of FORTRAN code. If the code was structured, this would also facilitate constructing a program flow graph.

A program flow graph is another valid method of analyzing the code. It would have worked in this case if the code had been written in a nice, structured manner. However, in many cases, the code as written is very convoluted and an adequate flow graph can not be constructed without a serious restructuring of the code. The algorithm detailed in Chapter Two provides a convenient methodology for constructing a program flow graph.

Probably the most important and productive task which could be done to continue this study would be to obtain the actual variable sizes. As stated earlier in the discussion, this task was not possible at this time because of all of the additional software that needs to be running in order to collect this data. This additional software is needed in order to supply the scene generator software with the

required scene data in the correct format. However, if this task were possible, it would provide the data to predict more exacting variable sizes.

The next task would be to find the execution path of the instructions. This task would also be simplified if the scene generator software were actually executing. If the scene generator programs were running, it would be possible to follow the execution through conditionals and loops by setting up intermediate output variables at all places in the code where a branch takes place. This information, along with a program flow graph, would provide a better look into the parallelism of the processes.

The third task would be to look at the functioning of the software itself. Since 525 lines must be produced in order to display one scene, there is a possibility that each line could be calculated in parallel. There is also a possibility that each pixel calculation would represent a separate identifiable calculation. Pixel calculations might be optimized by dedicating a separate computer configuration to this purpose.

Also, by analyzing the details of the software, it would be possible to streamline the executable code. This would involve removing all of the intermediate output such as printer output, instructions that calculate the timing of the major programs, and most of the disk input and output. Most of the disk input and output exists because of the way the software passes data between processes. This portion of the disk operations could be removed if all of the processes remained in memory at all times. Some of the disk operations would remain in order for the scene data itself to pass from data base to data base.

Finally, there is no need to restrict the design of the dynamic architecture computer to a few simple configurations. There are many more possible combinations that could be constructed using the same amount of hardware. These combinations should be further optimized with the additional data collected in the foregoing suggested steps.

It is also not necessary to limit the design of the dynamic architecture computer to a small set of hardware with a small set of configurations. Given a good program flow graph and adequate data on the size of all variables

involved, an architecture could be designed that would be as wide as necessary to execute all possible parallel paths. It is probable that the number of configurations that would be needed would still be limited.

## References

1. Dimond, K. R., and A. J. King. "A Flexible Development System for Microprogrammable Microprocessors," International Journal of Electrical Engineering Education, April 1979, pp. 156-165.
2. Estrin, Gerald. "Organization of Computer Systems: The Fixed Plus Variable Structure Computer," Proceedings of the Western Joint Computer Conference, 1960, pp. 33-40.
3. Estrin, Gerald, B. Bussell, R. Turn, J. Bibb. "Parallel Processing in a Restructable Computer System," IEEE Transactions on Electronic Computers, Vol. EC-12, pp. 747-755, 1963.
4. Fuchs, Henry, and Brian W. Johnson. "An Expandable Multiprocessor Architecture For Video Graphics," The Sixth Annual Symposium On Computer Architecture. New York: IEEE, 1979, pp.58-67.
5. Kartashev, Steven I., and Svetlana P. Kartashev. "A Powerful LSI Metacomputer System With Dynamic Architecture For Simulation Of Complex Problems," Modeling And Simulation Annual Pittsburgh Conference Proceedings. Pittsburgh: ISA, 1977, pp. 483-488.
6. Kartashev, Steven I., and Svetlana P. Kartashev. "Designing LSI Modular Computers And Systems," Proceedings Of The International Symposium On Mini And Micro Computing. New York: IEEE, November 1978, PP. 1-9.
7. Kartashev, Steven I., and Svetlana P. Kartashev. "Dynamic Architectures: Problems And Solutions," Computer, July 1978, pp. 26-40.
8. Kartashev, Steven I., and Svetlana P. Kartashev. "LSI Modular Computers, Systems, And Networks," Computer, July 1978, pp. 7-15.
9. Kartashev, Steven I., and Svetlana P. Kartashev. "Software Problems For Dynamic Architectures: Adaptive Assignment Of Hardware Resources," IEEE Computer Society International Computer Software And Application Conference. New York: IEEE, 1978, pp. 775-780.

10. Kartashev, Steven I., and Svetlana P. Kartashev. "Adaptable Pipeline System With Dynamic Architecture," Proceedings Of 1979 International Conference On Parallel Processing. New York: IEEE, 1979, pp. 222-230.
11. Kartashev, Steven I., and Svetlana P. Kartashev. "A Multicomputer System With Dynamic Architecture," IEEE Transactions On Computers, October 1979, pp. 704-721.
12. Kartashev, Steven I., and Svetlana P. Kartashev. "Distribution Of Programs For A System With Dynamic Architecture," IEEE Transactions On Computers, June 1982, pp. 488-514.
13. Kartashev, Steven I., and Svetlana P. Kartashev. "Adaptation Properties For Dynamic Architectures," AFIPS National Computer Conference Proceedings. Montvale: AFIPS Press, 1979, pp. 543-556.
14. Rauscher, Tomlinson G., and Ashok K. Agrawala. "Dynamic Problem-Oriented Redefinition Of Computer Architecture Via Microprogramming," IEEE Transactions On Computing, November 1978, pp. 1006-1014.
15. Vick, Charles R., Steven I. Kartashev, and Svetlana P. Kartashev. "Adaptable Architectures For Supersystems," Computer, November 1980, pp. 30-35.
16. Computer Program Product Specification, Mathematical Model for Scene Generation System CPCI Volume III, 20 Sept 1979, Contract F33657-78-C-0421.

## Bibliography

1. Kartashev, Steven I., and Svetlana P. Kartashev.  
"Evolution In Dynamic Architectures," Microprocessors  
And Microsystems, July-August 1979, pp. 249-250.
2. Wileden, Jack C. "An Introduction To the Modeling of  
Parallel Systems With Dynamic Structure," Proceedings of  
1979 International Conference On Parallel Processing,  
New York: IEEE, 1979, pp. 65-73.
3. White, Donnamaie E. Bit-Slice Design: Controllers and  
ALUs, New York: Garland STPM Press, 1981.
4. Myers, Glenford J. Digital System Design With LSI  
Bit-Slice Logic, New York: John Wiley and Sons, Inc.,  
1980.
5. Mick, John, and James Brick. Bit-Slice Microprocessor  
Design, New York: McGraw-Hill, 1980.
6. ---. The Am2900 Family Data Book, Sunnyvale CA: Advanced  
Micro Devices, 1979.

Appendix A  
Module Calling Summary

<u>Module</u>	<u>Calls</u>	<u>Called By</u>	<u>Main Program</u>
1 AREA1		LR2, SINGS, TB2	FRAME3
	COLOR		
2 AREA2		LR2, SINGS, TB2	FRAME3
	COLOR		
3 AREA3		TB2	FRAME3
	COLOR		
4 AREA4		LR2	FRAME3
	COLOR		
5 ARECAL		PTLGEN	FRAME3
6 AREMOD		PTLGEN	FRAME3
7 CMNOUT		SCGEN	SCGEN
	PUT		
	PUTCLR		
	PUTSET		
8 COL		FACPRO, PTCAL	FRAME2
	CPBLND		
	CPFADE		
	CPLITE		
9 COLOR		AREA1, AREA2, AREA3, AREA4, LR2, SINGS, TB2	FRAME3
10 CPBLND		COL	FRAME2
11 CPFADE		COL	FRAME2
12 CPLITE		COL	FRAME2
13 CSDEF		PARSEL	FRAME3
14 CXMAP		FRAME3	FRAME3
	CLOSE		
	PUT		
	PUTCLR		
	PUTSET		
	REED		
	SETFIL		
	SETRD		
15 DECODE		PRIRSV	FRAME3
16 DLCAL		FACPRO	FRAME2
	REED2		
	SETRD2		
	VTP		
17 DRCTRY		SCGEN	SCGEN
	EXIT		
	IPUT		
	REED		
	SETRD		

<u>Module</u>	<u>Calls</u>	<u>Called By</u>	<u>Main Program</u>
18 EDGCAL	EXIT FEP NEWED REED2 SETRD2 VTP	FACPRO	FRAME2
19 EDGGEN	CLOSE PUT PUTSET SETFIL	FRAME3	FRAME3
20 EDGORD	BCLR BSET MODIFY MODRD MODSET ORDER PUTCLR REED SETRD	FRAME3	FRAME3
21 EDWOUT	PUT2	FRAME2, NEWED, NEWPL	FRAME2
22 ERRRPT		MDCLR2, MODCLR, MODFY2, MODIFY, MODRD, MODSET, MODST2, PTCLR2, PUT, PUT2, PUTCLR, REED, REED2, SETRD, SETRD2	FRAME1 FRAME2 FRAME3 PRIPRO SCGEN FACCOM FACCOM
23 FACCOM	EXIT MODIFY MODRD MODSET SETFIL TIME		
24 FACOUT	PUT2	FACPRO	FRAME2
25 FACPRO	BSET COL DLCAL EDGCAL EXIT FACOUT FMOD MAPNDX PTCAL REED2 SETRD2 VTP	FRAME2	FRAME2

	<u>Module</u>	<u>Calls</u>	<u>Called By</u>	<u>Main Program</u>
26	FADCMP		FRAME1	FRAME1
27	FEP		EDGCAL,PTCAL	FRAME2
28	FMOD		FACPRO	FRAME2
29	FRAME1			FRAME1
		CLOSE		
		FADCMP		
		LNGLAT		
		LOD		
		MMFAD		
		MODCLR		
		MODIFY		
		MODSET		
		MOVE		
		MULT		
		PUT		
		PUTCLR		
		PUTSET		
		REED		
		ROTMAT		
		SETFIL		
		SETRD		
		TIME		
		TMULT		
		TRANS		
		TTMUL		
		TVEC		
		VEC		
		VTP		
		WINDOW		
30	FRAME2			FRAME2
		CLOSE		
		EDWOUT		
		FACPRO		
		HDROUT		
		INIT2		
		LSTOUT		
		MULT		
		PTCLR2		
		PUT2		
		REED2		
		SETRD2		
		TIME		
		TVEC		
		UPDATE		
		VTP		

	<u>Module</u>	<u>Calls</u>	<u>Called By</u>	<u>Main Program</u>
31	FRAME3	CXMAP EDGGEN EDGORD EXIT INIT3 ORDER PATPRO PRIHSV PTLGEN PTLSET RSTPED RSTPLT STPED STPLT TIME VIDOUT VIDPRO WNDDMP		FRAME3
32	HDROUT	PUT2	FRAME2	FRAME2
33	INIT2	CLOSE PUTST2 REED2 SETFIL SETRD2	FRAME2	FRAME2
34	INIT3	CLOSE RAMSET REED SETFIL SETRD	FRAME3	FRAME3
35	INPUT	REED SETRD	SCGEN	SCGEN
36	LNGLAT		FRAME1	FRAME1
37	LOD		FRAME1	FRAME1
38	LR2	EXIT	PARSEL	FRAME3
		AREA1 AREA2 AREA4 COLOR MODSELECT - Internal		
39	LSTOUT	PUT2	FRAME2	FRAME2
40	MAPNDX		FACPRO	FRAME2
41	MDCLR2		UPDATE	FRAME2
		ERRRPT EXIT SYSIO		

	<u>Module</u>	<u>Calls</u>	<u>Called By</u>	<u>Main Program</u>
42	MMFAD		FRAME1	FRAME1
43	MODCLR		FRAME1	FRAME1
		ERRRPT EXIT SYSIO		
44	MODCNT		PPFPL	PRIPRO
		PPCNT PPSORT		
45	MODFY2		UPDATE	FRAME2
		ERRRPT EXIT SYSIO		
46	MODIFY		EDGORD, FRAME1, PATPRO	FRAME1 FRAME3 FACCOM
		ERRRPT EXIT SYSIO		
47	MODRD		EDGORD, PATPRO	FRAME3 FACCOM
		ERRRPT EXIT SYSIO		
48	MODSET		EDGORD, FRAME1, PATPRO	FRAME1 FRAME3 FACCOM
		ERRRPT EXIT SYSIO		
49	MODST2		UPDATE	FRAME2
		ERRRPT EXIT SYSIO		
50	MODULA		PRIRSV	FRAME3
51	MOVE		FRAME1	FRAME1
52	MULT		FRAME1, FRAME2	FRAME1 FRAME2
53	NEWBLK		PPFPL	PRIPRO
		PPCNT RDBLK		
54	NEWED		EDGCAL	FRAME2
		BSET EDWOUT		
55	NEWPL		PTCAL	FRAME2
		EDWOUT		
56	NSEDGR		PRIRSV	FRAME3
57	NSOUT		NSRSLV	FRAME3
		PUT PUTSET SETFIL		
58	NSRSLV		PRIRSV	FRAME3
		NSOUT PRINTEDGETABLE - Internal		
59	ORDER		EDGORD, FRAME3	FRAME3
60	OVERID		PARSEL	FRAME3

	<u>Module</u>	<u>Calls</u>	<u>Called By</u>	<u>Main Program</u>
61	PARSEL	CSDEF LR2 OVERID SINGS TB2	PRIRSV	FRAME3
62	PATPRO	CLOSE MODIFY MODRD MODSET PRDMP PUTCLR	FRAME3	FRAME3
63	PPCNT		MODCNT,NEWBLK	PRIPRO
64	PPFPL		PRIPRO	PRIPRO
		MODCNT NEWBLK		
65	PPINP	CLOSE REED SETFIL SETRD	PRIPRO	PRIPRO
66	PPLIST		PRIPRO	PRIPRO
67	PPMSG		PRIPRO	PRIPRO
68	PPSORT		MODCNT,PPUOL	PRIPRO
69	PPUOL		PRIPRO	PRIPRO
		PPSORT		
70	PRAPLU		PRIRSV	FRAME3
71	PRAREA		PRIRSV	FRAME3
72	PRAUPD		PRELOD	FRAME3
73	PRCLR		PRIRSV	FRAME3
74	PRDMP		PATPRO	FRAME3
75	PREDGR		PRIRSV	FRAME3
76	PREEFS		PRELOD	FRAME3
77	PRELOD		PRIRSV	FRAME3
		PRAUPD PREEFS PRNEFS PRNXTO PRSTOR		
78	PREPD		PRIRSV	FRAME3
79	PRESEL		PRIRSV	FRAME3
80	PRFBKU		PRIRSV	FRAME3
81	PRINIT		PRIRSV	FRAME3
82	PRIPRO	CLOSE EXIT PPFPL PPINP PPLIST PPMSG PPUOL TIME WRTFPL		PRIPRO

<u>Module</u>	<u>Calls</u>	<u>Called By</u>	<u>Main Program</u>
83 PRIRSV	CLOSE DECODE MODULA NSEDGR NSRSLV PARSEL PRAPLU PRAREA PRCLR PREDGR PRELOD PREPD PRESEL PRFBKU PRINIT PROUT PRTPLU PRVIS SELDMP	FRAME3	FRAME3
84 PRNEFS		PRELOD	FRAME3
85 PRNXTO		PRELOD	FRAME3
86 PROUT		PRIRSV	FRAME3
	PUT PUTSET SETFIL		
87 PRSTOR		PRELOD	FRAME3
88 PRTPLU		PRIRSV	FRAME3
	CLEARTRANSITIONLIST - Internal		
89 PRVIS		PRIRSV	FRAME3
90 PTCAL		FACPRO	FRAME2
	COL FEP NEWPL REED2 SETRD2 VTP		
91 PTCLR2		FRAME2	FRAME2
	ERRRPT EXIT SYSIO		
92 PTLGEN		FRAME3	FRAME3
	ARECAL AREMOD CLOSE PUTCLR SAVELT		
93 PTLSIT		FRAME3	FRAME3
	CLOSE PUT PUTSET SETFIL		

Module	Calls	Called By	Main Program
94 PUT	ERRRPT EXIT SYSIO	CMNOUT, CXMAP, EDGGEN, FRAME1, NSOUT, PROUT, PTLSIT, SAVELT, WRTFPL	FRAME1 FRAME3 PRIPRO SCGEN
95 PUT2	ERRRPT EXIT SYSIO	EDWOUT, FACOUT, FRAME2, HDROUT, LSTOUT	FRAME2
96 PUTCLR	ERRRPT EXIT SYSIO	CMNOUT, CXMAP, EDGORD, FRAME1, PATPRO, PTLGEN, WRTFPL	FRAME1 FRAME3 PRIPRO SCGEN
97 PUTSET		CMNOUT, CXMAP, EDGGEN, FRAME1, NSOUT, PROUT, PTLSIT, SAVELT, WRTFPL	FRAME1 FRAME3 PRIPRO SCGEN
98 PUTST2		INIT2	FRAME2
99 RAMOUT		VIDOUT	FRAME3
	EXIT IOERR SYSIO		
100 RAMSET		INIT3	FRAME3
	EXIT IOERR SYSIO		
101 RDBLK		NEWBLK	PRIPRO
	REED SETFIL SETRD		
102 REED	ERRRPT EXIT SYSIO	CXMAP, DRCTRY, EDGORD, FRAME1, INIT3, INPUT, PPINP, RDBLK, RSTPED, RSTPLT, STPED, STPLT, TSTXMD, VPAINC, VPLTC, WNDDMP	FRAME1 FRAME3 PRIPRO SCGEN
103 REED2	ERRRPT EXIT SYSIO	DLCAL, EDGCAL, FACPRO, FRAME2, INIT2, PTCAL	FRAME2
104 ROTMAT		FRAME1	FRAME1
105 RSTPED		FRAME3	FRAME3
	CLOSE REED		
106 RSTPLT		FRAME3	FRAME3
	CLOSE REED		
107 SAVELT		PTLGEN	FRAME3
	PUT PUTSET SETFIL		

	<u>Module</u>	<u>Calls</u>	<u>Called By</u>	<u>Main Program</u>
108	SCGEN	CLOSE CMNOUT DRCTRY INPUT SETFIL TIME		SCGEN
109	SELDMP		PRIRSV	FRAME3
110	SETFIL	EXIT OPENW	CXMAP, EDGGEN, FACCOM, FRAME1, INIT2, INIT3, NSOUT, PPINP, PROUT, PTLSIT, SAVELT, SCGEN, STPED, STPLT, UPDATE, VPAINC, VPLTC, WNDDMP, WRTFPL	FRAME1 FRAME2 FRAME3 PRIPRO SCGEN FACCOM
111	SETRD	ERRRPT EXIT SYSIO	CXMAP, DRCTRY, EDGORD, FRAME1, INIT3, INPUT, PPINP, RDBLK, STPED, STPLT, TSTXMD, VPAINC, VPLTC, WNDDMP	FRAME1 FRAME3 PRIPRO SCGEN
112	SETRD2	ERRRPT EXIT SYSIO	DLCAL, EDGCAL, FACPRO, FRAME2, INIT2, PTCAL	FRAME2
113	SINGS	AREA1 AREA2 COLOR MODSELECT - Internal	PARSEL	FRAME3
114	STPED	CLOSE REED SETFIL SETRD	FRAME3	FRAME3
115	STPLT	CLOSE REED SETFIL SETRD	FRAME3	FRAME3
116	TB2	AREA1 AREA2 AREA3 COLOR MODSELECT - Internal	PARSEL	FRAME3
117	TMULT		FRAME1	FRAME1
118	TRANS		FRAME1	FRAME1
119	TSBNST		TSBSNO	FRAME3
120	TSBSNO		TSESP	FRAME3
121	TSDBN	TSBNST	TSESP	FRAME3

	<u>Module</u>	<u>Calls</u>	<u>Called By</u>	<u>Main Program</u>
122	TSEA		TSEDA	FRAME3
123	TSEDA		TSESP	FRAME3
		TSEA		
124	TSEDGR		TSESP, TSINIT	FRAME3
125	TSEMOV		TSESP	FRAME3
126	TSESP		VPTEX	FRAME3
		TSBSNO		
		TSDBN		
		TSEDA		
		TSEDGR		
		TSEMOV		
		TSLODS		
		TSMUX		
		TSPINC		
		TSSHAD		
		TSTXMD		
127	TSINIT		VPTEX	FRAME3
		TSEDGR		
128	TSLOD		TSLODS	FRAME3
129	TSLODS		TSESP	FRAME3
		TSLOD		
130	TSMUX		TSESP	FRAME3
131	TSPINC		TSESP	FRAME3
132	TSSHAD		TSESP	FRAME3
133	TSTXMD		TSESP	FRAME3
		REED		
		SETRD		
		SETUPLD	- Internal	
		SETUPMAP	- Internal calls REED and SETRD	
134	TTMUL		FRAME1	FRAME1
135	TVEC		FRAME1, FRAME2	FRAME1
				FRAME2
136	UPDATE		FRAME2	FRAME2
		CLOSE		
		MDCLR2		
		MODFY2		
		MODST2		
		SETFIL		
137	VEC		FRAME1	FRAME1
138	VIDOUT		FRAME3	FRAME3
		RAMOUT		
139	VIDPRO		FRAME3	FRAME3
		CLOSE		
		VPAINC		
		VPCFC		
		VPFADE		
		VPIFLD		
		VPILN		
		VPLNDL		
		VPLTC		
		VPMLF		
		VPSIMP		
		VPTEX		

	<u>Module</u>	<u>Calls</u>	<u>Called By</u>	<u>Main Program</u>
140	VPAINC		VIDPRO	FRAME3
		REED SETFIL SETRD		
141	VPCFC		VIDPRO	FRAME3
142	VPFADE		VIDPRO	FRAME3
143	VPIFLD		VIDPRO	FRAME3
144	VPILN		VIDPRO	FRAME3
145	VPLNDL		VIDPRO	FRAME3
146	VPLTC		VIDPRO	FRAME3
		CLOSE REED SETFIL SETRD		
147	VPMLF		VIDPRO	FRAME3
148	VPSIMP		VIDPRO	FRAME3
149	VPTEX		VIDPRO	FRAME3
		TSESP TSINIT		
150	VTP		DLCAL, EDGCAL, FACPRO, FRAME1, FRAME2, PTCAL	FRAME1 FRAME2
151	WINDOW		FRAME1	FRAME1
152	WNDDMP		FRAME3	FRAME3
		REED SETFIL SETRD		
153	WRTFPL		PRIPRO	PRIPRO
		PUT PUTCLR PUTSET SETFIL		

#### System Modules

<u>Name</u>	<u>Called By</u>
BCLR	EDGORD
BSET	EDGORD, FACPRO, NEWED
CLOSE	CXMAP, EDGGEN, FRAME1, FRAME2, INIT2, INIT3, PATPRO, PPINP, PRIPRO, PRIHSV, PTLGEN, PTLSET, RSTPED, RSTPLT, SCGEN, STPED, STPLT, UPDATE, VIDPRO, VPLTC

<u>Name</u>	<u>Called By</u>
EXIT	DRCTRY, EDGCAL, FACCOM, FACPRO, FRAME3, LOD, MDCLR2, MODCLR, MODFY2, MODIFY, MODRD, MODSET, MODST2, PRIPRO, PTCLR2, PUT, PUT2, PUTCLR, RAMOUT, RAMSET, REED, REED2, SETFIL, SETRD, SETRD2
IOERR	RAMOUT, RAMSET
IPUT	DRCTRY
OPENW	SETFIL
SYSIO	MDCLR2, MODCLR, MODFY2, MODIFY, MODRD, MODSET, MODST2, PTCLR2, PUT, PUT2, PUTCLR, RAMOUT, RAMSET, REED, REED2, SETRD, SETRD2
TIME	FACCOM, FRAME1, FRAME2, FRAME3, PRIPRO, SCGEN

Appendix B  
Module Descriptions

<u>Module</u>	<u>Description</u>
1 AREA1	generates the scene object areas for area 1 specified by the edge parameter tables included in the subfunctions SINGS, TB2, and LR2.
2 AREA2	generates the scene object areas for area 2 specified by the edge parameter tables included in the subfunctions SINGS, TB2, and LR2.
3 AREA3	generates the scene object areas specified by the edge parameters in the two-edge top/bottom case.
4 AREA4	generates the scene object areas for area 4 specified in subfunction LR2.
5 ARECAL	calculates areas subtended by a point source for a given raster line.
6 AREMOD	modifies the light source area by comparing two point lights within that area.
7 CMNOUT	creates a disk file that contains the intermodule COMMON data. This data file serves as the intermodule data link between all the modules of the Scene Generator. The PUT submodule is the primary means of dumping the COMMON's out to disk.
8 COL	processes, per entry, one face or light. For normal faces, it builds and modifies for sun illumination; for light faces, it modifies color by brightness; for point lights, it calculates fading range; and it stores results in modified color memory.
9 COLOR	generates the color intensity parameters specified by the case parameter tables.
10 CPBLND	integrates into a face either sky, ground, haze or the next indicated face. Once done, this module will then modify the brightness of the face according to the sun illumination.
11 CPFAD	calculates the fading coefficient depending upon range and the fading determinate. It also returns arguments for full fading and for no fading.
12 CPLITE	determines brightness, color, and size for point lights using light parameter constants, the light extinguishing curve and range.
13 CSDEF	identifies the conditions which determine two-edge case numbers and sets an indicator to the code representing the extant conditions.
14 CXMAP	constructs a table in which the number of edge vertices on each raster line are recorded and the number of point source boundaries that start and stop on each raster line are recorded.

<u>Module</u>	<u>Description</u>
15 DECODE	examines the face characteristics and edge flags of the four edges in the edge load store to decode the case types.
16 DLCAL	checks to see if designated light is in universal features file; reads data from disk using SETRD2 and REED2; checks for visibility of light.
17 DRCTRY	orders the Environment Data Blocks (EDB's) that make up the Environment Data Base; pull all of the data out of the EDB's that will be used by the Frame I Module. A directory of pointers is constructed; each pointer is the actual record number of an EDB in the data base. The result is an ordered list of pointers that organize the EDB's by coarse region centroid, and by level of detail within each coarse region. The number of 12' x 12' regions in the data base, along with their centroids and their coordinate set indicator is determined.
18 EDGCAL	distinguishes between environmental and universal data, inputs correct data, rotates vertices as necessary, sets up appropriate variables, processes face edges, builds edges of lights, and stores data through NEWED submodule.
19 EDGGEN	calculates the left and right intercepts where each edge crosses the raster line.
20 EDGORD	processes an ordered list of edge left intercepts to obtain a list of relative face numbers, and from that list, a list of relative priority numbers for the faces in the scene.
21 EDWOUT	counts the number of edges in the face and the number of lights, supplies the proper headings, and, using submodule PUT2, puts data into files corresponding to disk sector size.
22 ERRRPT	reports any error received as the result of an input/output operation.
23 FACCOM	compresses face list.
24 FACOUT	determines face type, modifies terrain face based upon priority range, calculates feature numbers for universal objects, and calculates relative face number and writes the record into the active face list.
25 FACPRO	clears active external face list, checks on the number of faces to be processed, reads data from buffer, computes values, checks for visibility and universal features, computes variables dependent upon sun illumination, checks texture orientation, builds temporary active face list, checks for proper organization of data and builds new external face list.

<u>Module</u>	<u>Description</u>
26 FADCMP	calculates the Frame II and Frame III fading and horizon coefficients to be used in sky, ground and haze color processing.
27 FEP	tests to determine if the mode is point light or face. If face it then tests for minimum size light face. If it is minimum size light or point light, a test is made to determine if the light is visible in channel is tested. If not minimum size light face or point light, the vertices are checked to determine if any boundaries have been crossed, and if so, the vertices are replaced and tested. Then it tests for a possible pseudo edge and defines it if necessary.
28 FMOD	computes the pattern/shading coefficients for Frame III by computing the view point vector, rotating face vectors as needed, retrieving the appropriate texture/shading data and determining texture coefficients.
29 FRAME1	calculates the rotation matrices needed by other modules, performs region channel assignment on environment data provided by the Scene Generator Controller Module, and calculates fading and horizon coefficients for fading; performs the executive function of controlling the other submodules within the task FRAME1.
30 FRAME2	initializes, reads subregion data and processes clusters, faces and universal features and updates the intermodule COMMON data file for use by future modules; performs the executive function of controlling the other submodules within the task FRAME2.
31 FRAME3	calls the subfunctions that constitute the color intensity calculations in the required sequence; performs the executive function of controlling the other submodules within the task FRAME3.
32 HDROUT	generates and then writes the header to the active face list. The header will be comprised of the BLOCK#, BLOCK TYPE, VP VECTOR, and the BLOCK SCALING FACTOR.
33 INIT2	initializes files, builds logical unit table to match with other input/output submodules, reads in COMMONS, opens all Frame II files and initializes the test green color table.
34 INIT3	clears appropriate COMMON areas, opens the necessary input files stored on disk and reads selected input data.

<u>Module</u>	<u>Description</u>
35 <u>INPUT</u>	reads three data files needed by the Scene Generator Modules: the Visual Parameter file, the Environment Data Base Header file and the Color/Light Parameter file. These files are used to build the intermodule COMMON's that provide other modules with necessary data. Using the REED submodule, actual disk accesses are transparent to the INPUT submodule, allowing data to be taken from the disk files in smaller blocks only as needed.
36 LNLAT	creates a nadir to geocentric rotation matrix based on longitude and latitude and converts the viewpoint from longitude, latitude. and altitude to feet from the earth's center.
37 LOD	selects data blocks to be processed based on level of detail and coarse region.
38 LR2	processes two-edge left/right cases as decoded by subfunction DECODE. Processing will be of edge set parameters.
39 LSTOUT	writes a last face record to the active face list.
40 MAPNDX	calculates the proper index into the AMAP and NAMAP arrays, based upon face number.
41 MDCLR2	is the third part of a three part group that modifies the data in a disk file. This part closes out those buffers and files that the first two parts may have used.
42 MMFAD	calculates the 3-D fading coefficients and colors for all moving models in the data base.
43 MODCLR	closes out the buffers and files that have been modified by submodule MODIFY.
44 MODCNT	computes the model/object count for 3-D face group; generates a list of model numbers in ascending order and computes object counts for each model in the group.
45 MODFY2	second of a three part group to modify data in a disk file. From the information set up by MODST2 (index, absolute address, record number and the data in the record), this submodule will modify, clean up data, store in as many buffers as needed, and then write them to the correct data disk file.
46 MODIFY	modifies a data file by reading in a buffer of data, modifying it, and writing it back out to the disk at its originally read location.
47 MODRD	performs the necessary calculations to determine how much data to be modified should be moved from the I/O buffer to the buffer used to modify the data.

<u>Module</u>	<u>Description</u>
48 MODSET	modifies disk resident data in the same way main memory resident data is modified.
49 MODST2	is the first of a three-part group of modules that will modify data in a disk file. This module checks for index out of bounds, determines absolute address, gets the record number, reads the file and stores it for MODIFY2.
50 MODULA	sets the modulation and fading select codes for each of the three colors in the visible edge data set.
51 MOVE	moves one matrix into another.
52 MULT	multiplies two matrices and returns the result in a third matrix.
53 NEWBLK	initializes pointer variables and reads in a new priority data block when the active block number is changed; determines the model counts for all the models in the block.
54 NEWED	determines model number, object number, terrain face flags, computes edge control word, puts it into temporary buffer, and arranges data to fit temporary active face list.
55 NEWPL	adds a new point light to the appropriate edge data word and updates necessary control files.
56 NSEDGR	moves an edge data set from common areas GEN and EDREL to common area NSEDGE.
57 NSOUT	moves data from common area NSTABL to common area PRVP.
58 NSRSLV	receives edges one at a time from NSEDGR and puts them in a table based on priority-right.
59 ORDER	generates a list by ordering incoming values in ascending order.
60 OVERID	overrides the case type results from the DECODE subfunction when valid edge, collapsed edge, and certain flag conditions are met.
61 PARSEL	controls the overall selection process of choosing edge parameters.
62 PATPRO	calculates delta I - JN dependent coefficients for output to the video processor.
63 PPCNT	processes separation plane data and generates the counts for a specified group.
64 PPFPL	assigns the absolute face priority numbers based on data from the active face list, active model list, used model/overlay numbers list, the priority data memory and the universal objects relative priority list.
65 PPINP	interfaces between data as stored on disk and as needed by the FRAME2 and FRAME3 submodules through the priority processor submodules; saves priority information for universal features.

<u>Module</u>	<u>Description</u>
66 PPLIST	creates various lists by cycling through the active face list and recording the appropriate information; generates the highest priority count for faces at given terrain face values.
67 PPMSG	displays a specified error message.
68 PPSORT	generates an ordered list of keys so that the corresponding values are in ascending order.
69 PPUOL	creates the universal objects relative priority list used in assigning absolute priority numbers in submodule PPFPL.
70 PRAPLU	updates the next active priority list based on the next ordered edge data for the current raster line.
71 PRAREA	calculates the area in the raster line element to the right of the edge whose J-left intercept intersects a top or bottom boundary of the element.
72 PRAUPD	updates the element area calculation for each valid edge in the edge load store.
73 PRCLR	clears the memory pointers and flags.
74 PRDMP	writes out the data from designated common areas generated in FRAME II.
75 PREDGR	dissects the edge flag word and stores the unpacked flags and data in individual data words contained in common area PREDGR.
76 PREEFS	processes edges tagged as an equal edge by the edge selection.
77 PRELOD	cycles through the top two priority level of edges in the edge select memory.
78 PREPD	determines the two or three highest priority levels extant in the active and transition priority list.
79 PRESEL	selects an additional edge which intersects the current raster line element for each priority level currently retained in the transition priority list.
80 PRFBKU	simulates the read-write function of the fallback memory.
81 PRINIT	clears the processing flags and main memory areas.
82 PRIPRO	cycles through the active face list to form the used overlay numbers/models numbers list, the active models list, the active universal objects range/count lists, and counts the number of faces at each possible terrain face range value by use of other submodules.
83 PRIRSV	calls the submodules that constitute the priority resolver process.
84 PRNEFS	categorizes the non-equal edge sets in the edge load store into three cases for modification.

<u>Module</u>	<u>Description</u>
85 PRNXTO	performs face modification for equal A edges and equal B edges and then ascertains if the A edges are next to the top/bottom of the raster line.
86 PROUT	gathers data to be stored in common area PRVP for later use by the video processor.
87 PRSTOR	transfers the edge data to be processed from the edge select memory to the edge load store memory.
88 PRTPLU	selects from a large number of edges the eight best edges that intersect the current raster line at a single element.
89 PRVIS	prints the contents of the common area PRLD.
90 PTCAL	obtains point light relative addresses, light characteristics, number of lights per string, rotates light vertex to proper window, and adds new point light to edge data word file.
91 PTCLR2	outputs and clears buffer.
92 PTLGEN	simulates the point light generator for use with the camera station.
93 PTLSIT	retrieves FRAME2 point light data from the disk and places it in a temporary disk file.
94 PUT	writes data out to a disk file in such a way that the disk access is transparent to the calling program.
95 PUT2	transfers data to system disk after arranging data to exactly fill a disk sector.
96 PUTCLR	clears the output buffer to the disk file.
97 PUTSET	clears the output buffer to be filled by subsequent calls from the PUT submodule.
98 PUTST2	sets a pointer to the beginning of the storage buffer, checks to see if all locations in the buffer have been processed, and if so, clears the entire buffer.
99 RAMOUT	transfers color intensity data generated in the FRAME III process to buffers and then to the display device.
100 RAMSET	initializes the display device.
101 RDBLK	reads in a new priority data block according to block type and block number.
102 REED	regulates the reading of data off of a disk file by calling the system subroutine SYSIO and keeping track of and updating the sector and word pointers.
103 REED2	reads data supplied from SETRD2 into buffers equal in size to a disk sector.
104 ROTMAT	creates a direction cosine matrix via attitude rotation.
105 RSTPED	retrieves a number of edge data word sets for processing by the Edge Generator submodule.

<u>Module</u>	<u>Description</u>
106 RSTPLT	retrieves point light data words from a buffer and stores them in common.
107 SAVELT	takes point light data from a common area and transfers the data to another area for later storage.
108 SCGEN	sets up the necessary data files and organizes the environment data for the other modules; performs the executive function of controlling the other modules in the task.
109 SELDMP	dumps all values of the module's variables to the line printer.
110 SETFIL	sets up the disk data files by using the system subroutine OPENW to open the files.
111 SETRD	sets up the necessary pointers to begin reading the appropriate disk file; performs the initial read; saves the sector number and relative word address in the sector for subsequent calls by the submodule REED.
112 SETRD2	determines an absolute address based upon relative address and resolution; reads the file into a buffer.
113 SINGS	processes single edge cases as decoded by subfunction DECODE.
114 STPED	retrieves a number of edge data word sets for processing by the Edge Generator submodule.
115 STPLT	retrieves point light data word sets for processing by the point light generator subroutine.
116 TB2	processes two-edge top/bottom cases as decoded by subfunction DECODE.
117 TMULT	multiplies a transposed matrix by a second matrix and returns the result in a third matrix.
118 TRANS	transposes a matrix and returns the result in another matrix.
119 TSBNST	determines texture/shading base number set type.
120 TSBSNO	calculates texture/shading base number calculation.
121 TSDBN	calculates base number per element change.
122 TSEA	calculates texture element area.
123 TSEDA	detects edge of texture area.
124 TSEDGR	reads next edge into texture/shading routine.
125 TSEMOV	moves pattern word data to current edge common area.
126 TSESP	performs all texture/shading calculations for an element set.
127 TSINIT	initializes texture/shading calculations at start of line.
128 TSLOD	calculates level of detail.
129 TSLODS	selects texture level of detail.

<u>Module</u>	<u>Description</u>
130 TSMUX	multiplexes texture/shading output.
131 TSPINC	generates texture pattern-incrementer output.
132 TSSHAD	processes shading information.
133 TSTXMD	processes texture modulation, smoothing, and summation functions.
134 TTMUL	multiplies two transposed matrices and returns the result in a third matrix.
135 TVEC	multiplies a transposed matrix and a vector and returns the result in a second vector.
136 UPDATE	updates the common data file by using the interval submodule SETFIL and modifying the data for use by FRAME3.
137 VEC	multiplies a matrix and a vector and stores the results in a second vector.
138 VIDOUT	invokes the submodule RAMOUT to supply data to the display device for each raster line as the processing for that line is completed.
139 VIDPRO	calls the subfunctions that constitute the video processor routine.
140 VPAINC	compresses the colors and subtended areas of two edge data functions, A and B for each line element over the interval that the given edge word is active; a third color C is included for the remaining area.
141 VPCFC	combines face colors using current element areas.
142 VPFADE	determines coefficients, horizon flags, and multiplies 3 areas in current element.
143 VPIFLD	initializes the fading range for ground and sky for the upper left corner, transferred only per field line.
144 VPILN	updates pointers and resets fade ranges for a new line, and determines horizon flag.
145 VPLNDL	simulates the directional illumination envelope associated with landing lights.
146 VPLTC	retrieves the color and area of any point light in current element.
147 VPMLF	merges light colors with face colors and puts final color into output line buffer.
148 VPSIMP	processes the simplified video processor functions.
149 VPTEX	provides the interface between the video processor and the texture generator.
150 VTP	performs channel assignment on faces, clusters, and regions and rotates vertices; in Frame I it is used to determine whether regions/subregions will be visible.

	<u>Module</u>	<u>Description</u>
151	WINDOW	calculates the window boundary constants used to determine whether data will be visible in the view window.
152	WNDDMP	dumps header data for each edge crossing within user specified values.
153	WRTFPL	dumps the absolute face priority list to an output file.

Appendix C  
Collected Data on Variables

## Appendix C

The data in this appendix represents the data collected regarding the type and quantity of each type of variable in each of the six main programs. It is arranged in six tables. Each table has six columns as follows:

SYMBOL	name of the variable as found in the programs
T	indicates the type of variable as follows: C Character I Integer L Logical R Real
S	size of the variable in bytes
DIMN	dimension of the variable
LOCATI	location of the variable in memory. This will be the name of a COMMON block or a program if it is local.
TOTAL	total memory required for the storage of this variable

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
BUFF	I	4	320	BFRM	1280
D.AA	I	4	1	FACCOM	4
D.AA	I	4	1	MODSET	4
FILE1	R	8	1	FACCOM	8
FILE2	R	8	1	FACCOM	8
I	I	4	1	FACCOM	4
I	I	4	1	MODSET	4
IABSAD	I	4	1	MODSET	4
IARG	I	4	1	MODSET	4
IARG	I	4	1	SETFIL	4
IB	I	4	3600	FACCOM	14400
IC	I	4	1	FACCOM	4
ICON	I	4	1	FACCOM	4
ID	I	4	1200	FACCOM	4800
IELAP	I	4	1	FACCOM	4
IFACN	I	2	4096		8192
IFACT	I	4	1		4
IFN	I	4	1	FACCOM	4
IMIN	I	4	1	FACCOM	4
INDAFN	I	2	10000	FACCOM	20000
INX	I	4	1	FACCOM	4
IPROC	I	4	1	MODSET	4
IREC	I	4	5	BFRM	20
IRFC	I	4	1	MODSET	4
IRX	I	4	1	MODSET	4
ISEC	I	4	1	FACCOM	4
ISTAT	I	4	1	MODSET	4
ISTAT	I	4	1	SETFIL	4
ISTIM	I	4	3	FACCOM	12
IT	I	4	1	FACCOM	4
ITIM	I	4	3	FACCOM	12
IUP	I	4	1	MODSET	4
IX	I	4	5	BFRM	20
J1	I	4	1	MODSET	4
JARG	I	4	1	MODSET	4
JPROC	I	4	1	MODSET	4
JREC	I	4	1	MODSET	4
JRFC	I	4	1	MODSET	4
JX	I	4	1	MODSET	4
K	I	4	1	FACCOM	4
K	I	4	1	MODSET	4
KARG	I	4	1	MODSET	4
LARG	I	4	1	MODSET	4
LEFT	I	4	1	MODSET	4
LPCT	I	4	1	MODSET	4
M	I	4	1	FACCOM	4
MSKE	I	4	1	FACCOM	4
MSKL	I	4	1	FACCOM	4
N	I	4	1	MODSET	4
NMED	I	4	1	FACCOM	4
NNSEC	I	4	1	FACCOM	4
NOEDG	I	4	1	FACCOM	4
NOSET	I	4	1	FACCOM	4

Table C-1: Variable List  
for FACCOM  
(Sheet 1 of 2)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
NOW	I	4	1	FACCOM	4
NSEC	I	4	1	FACCOM	4
RSLTN	I	4	6	MODSET	24
TBLK	I	4	5	MODSET	20
XXX	R	4	1	FACCOM	4

Table C-1: Variable List  
for FACCOM  
(Sheet 2 of 2)

TOTAL		
SIZE		236
DIMN		19,290
TOTAL		48,976

SYMBOL	T	S	DIMN	LOCATI	TOTAL
-----	-	-	----	-----	-----
A	I	4	1	LNGLAT	4
ACTREG	I	4	281	ABLIST	1124
AK	R	4	1	FIXDT	4
ATT	R	4	3	FRM1	12
AZIM	R	4	1	FRM1	4
BLKAMT	I	4	8	LOD	32
BLNFLG	L	4	1	OPTNS	4
BUFF	I	4	320	BFRI	1280
BUFF	I	4	320	BFRM	1280
BUFF	I	4	320	BFRO	1280
C	I	4	3	MMFAD	12
C	I	2	1	VTP	2
C1	R	4	1	FIXDT	4
C1	R	4	1	ROTMAT	4
C2	R	4	1	FIXDT	4
C2	R	4	1	ROTMAT	4
C3	R	4	1	FIXDT	4
C3	R	4	1	ROTMAT	4
C4	R	4	1	FIXDT	4
CI	R	4	1	WINDOW	4
CJ	R	4	1	WINDOW	4
CLC	I	4	18	FR1D	72
CNV	R	4	1	FRAME1	4
COLOR	R	4	768	TABLS	3072
COS	R	4	1	LNGLAT	4
CSI	I	2	6	FR1D	12
CTHEP	R	4	1	VTP	4
CTHETA	R	4	1	VTP	4
CV	R	4	1	FRM1	4
CW	R	4	1	FRM1	4
D	R	8	1	FRAME1	8
D	R	8	1	LNGLAT	8
D.AA	I	4	1	FADCMP	4
D.AA	I	4	1	FRAME1	4
D.AA	I	4	1	LOD	4
D.AA	I	4	1	MMFAD	4
D.AA	I	4	1	MODSET	4
D.AA	I	4	1	MULT	4
D.AA	I	4	1	PUT	4
D.AA	I	4	1	VTP	4
D.AB	I	4	1	FADCMP	4
D.AB	I	4	1	MULT	4
D.BA	I	4	1	FRAME1	4
D.BA	I	4	1	LNGLAT	4
D.BA	I	4	1	MULT	4
D.BB	I	4	1	FRAME1	4
D.BB	I	4	1	MULT	4
D.BC	I	4	1	FRAME1	4
D.CA	I	4	1	FRAME1	4
D.CA	I	4	1	MULT	4
D.CB	I	4	1	FRAME1	4
D.CC	I	4	1	FRAME1	4
DF	R	4	1	MMFAD	4

Table C-2: Variable List  
for FRAME1  
(Sheet 1 of 10)

SYMBOL	T	S	DIMN	LOCATI	TOTAL
-----	-	-	----	-----	-----
DF1B	R	4	1	FADE	4
DF1T	R	4	1	FADE	4
DF2	R	4	1	FADE	4
DF2MAX	R	4	1	FADCMP	4
DFG	R	4	1	FADCMP	4
DFP	R	4	1	CPFM	4
DFS	R	4	1	FADCMP	4
DIR	I	2	1686	DRCT	3372
E	R	4	1	FRAME1	4
E	R	4	1	LNGLAT	4
EDGFLG	L	4	1	OPTNS	4
ELEV	R	4	1	FRM1	4
EOF	I	4	1	REED	4
ERRMSG	I	4	1	MISC	4
F	I	2	1	VTP	2
F1RFLG	L	4	1	OPTNS	4
FADFLG	L	4	1	OPTNS	4
FILE	R	8	2	FRAME1	16
FOPG	R	4	1	VPFM	4
FOPS	R	4	1	VPFM	4
FR1EDB	I	4	2304	FR1D	9216
FVPG	R	4	1	VPFM	4
FVPS	R	4	1	VPFM	4
FWPG	R	4	1	VPFM	4
FWPS	R	4	1	VPFM	4
GND	I	2	3	VPFM	6
HAZG	I	2	3	VPFM	6
HAZS	I	2	3	VPFM	6
HF	R	4	1	WINDOW	4
HFOV	R	4	1	FRM1	4
I	I	4	1	FRAME1	4
I	I	4	1	LNGLAT	4
I	I	4	1	MODSET	4
I	I	4	1	MULT	4
I	I	4	1	PUT	4
I	I	4	1	REED	4
I	I	4	1	VTP	4
IO	I	4	1	FRM1	4
IOP	R	4	1	WINDOW	4
IA	I	4	1	ABLIST	4
IABSAD	I	4	1	MODSET	4
IABSAD	I	4	1	SETRD	4
IAFW	I	4	1	FRAME1	4
IARG	I	4	1	LOD	4
IARG	I	4	1	MODSET	4
IARG	I	4	1	PUT	4
IARG	I	4	1	REED	4
IARG	I	4	1	SETFIL	4
IARG	I	4	1	SETRD	4
IBEG	I	4	1	FRAME1	4
IBFLG	L	4	1	VTPDT	4
IBNUM	I	4	1	FRAME1	4
ICHAN	I	4	1	OPTNS	4

Table C-2: Variable List  
for FRAME1  
(Sheet 2 of 10)

SYMBOL	T	S	DIMN	LOCATI	TOTAL
ICHASS	L	4	1	VTPDT	4
ICHFLG	L	4	1	VTPDT	4
ICOSYS	I	4	1	MISC	4
ICT	I	4	1	LOD	4
IEADUF	I	4	1	UNFDT	4
IEF	I	4	1	MISC	4
IELAP	I	4	1	FRAME1	4
IEND	I	4	1	FRAME1	4
IFADUF	I	4	1	UNFDT	4
IFOGC	I	4	3	CPFM	12
IFXLOD	I	4	1	OPTNS	4
IGNDC	I	4	3	CPFM	12
IHAZC	I	4	3	CPFM	12
IHLD	I	2	128	FRAME1	256
IK	I	4	1	VTP	4
ILOD	I	4	1	VTPDT	4
IMIN	I	4	1	FRAME1	4
IMODEL	I	4	1	VTPDT	4
INCT	I	4	6	VTPDT	24
IPA	I	4	1	FRAME1	4
IPB	I	4	1	FRAME1	4
IPROC	I	4	1	MODSET	4
IPROC	I	4	1	PUT	4
IPROC	I	4	1	REED	4
IPROC	I	4	1	SETRD	4
IRC	I	4	1	FR1D	4
IREC	I	4	5	BFRI	20
IREC	I	4	5	BFRM	20
IREC	I	4	5	BFRO	20
IRFC	I	4	1	MODSET	4
IRFC	I	4	1	PUT	4
IRFC	I	4	1	REED	4
IRFC	I	4	1	SETRD	4
IRX	I	4	1	MODSET	4
IRX	I	4	1	SETRD	4
ISEC	I	4	1	FRAME1	4
ISKYC	I	4	3	CPFM	12
ISPF	I	4	1	VTP	4
ISPL	I	4	1	VTP	4
ISTAT	I	4	1	FRAME1	4
ISTAT	I	4	1	MODSET	4
ISTAT	I	4	1	PUT	4
ISTAT	I	4	1	REED	4
ISTAT	I	4	1	SETFIL	4
ISTAT	I	4	1	SETRD	4
ISTIM	I	4	3	FRAME1	12
ITADUF	I	4	1	UNFDT	4
ITEMP	I	4	1	FRAME1	4
ITIM	I	4	3	FRAME1	12
IU	I	4	1	UNFDT	4
IUP	I	4	1	MODSET	4
IUP	I	4	1	PUT	4
IUP	I	4	1	REED	4

Table C-2: Variable List  
for FRAME1  
(Sheet 3 of 10)

SYMBOL	T	S	DIMN	LOCATI	TOTAL
IX	I	4	5	BFRI	20
IX	I	4	5	BFRM	20
IX	I	4	5	BFRO	20
J	I	4	1	FRAME1	4
J	I	4	1	LNGLAT	4
J	I	4	1	LOD	4
J	I	4	1	MULT	4
JO	I	4	1	FRM1	4
JOP	R	4	1	WINDOW	4
J1	I	4	1	MODSET	4
J1	I	4	1	PUT	4
J1	I	4	1	REED	4
JARG	I	4	1	MODSET	4
JARG	I	4	1	PUT	4
JARG	I	4	1	REED	4
JARG	I	4	1	SETRD	4
JCT	I	4	1	LOD	4
JEL	I	4	1	JWIN	4
JER	I	4	1	JWIN	4
JPROC	I	4	1	MODSET	4
JREC	I	4	1	MODSET	4
JREC	I	4	1	REED	4
JREG	I	4	1	FRAME1	4
JRFC	I	4	1	MODSET	4
JSSW	I	4	1	MISC	4
JX	I	4	1	MODSET	4
JX	I	4	1	PUT	4
JX	I	4	1	REED	4
K	I	4	1	FADCMP	4
K	I	4	1	FRAME1	4
K	I	4	1	MMFAD	4
K	I	4	1	MODSET	4
K	I	4	1	MULT	4
K	I	4	1	PUT	4
K	I	4	1	REED	4
KA	I	4	1	LOD	4
KARG	I	4	1	MODSET	4
KARG	I	4	1	PUT	4
KGND	R	4	1	VPFM	4
KI	R	4	1	FRM1	4
KIJ	R	4	4	FIXDT	16
KIMIO	R	4	1	FIXDT	4
KIPIO	R	4	1	FIXDT	4
KJ	R	4	1	FRM1	4
KJMJO	R	4	1	FIXDT	4
KJPJO	R	4	1	FIXDT	4
KL	R	4	3	VTPDT	12
KLDF	R	4	1	VTPDT	4
KLDREZ	R	4	1	VTP	4
KLE	R	4	1	VTPDT	4
KLM	R	4	1	VTPDT	4
KLMREP	R	4	1	VTP	4
KLOD	I	4	1	FIXDT	4

Table C-2: Variable List  
for FRAME1  
(Sheet 4 of 10)

SYMBOL	T	S	DIMN	LOCATI	TOTAL
KLOR	R	4	1	VTP	4
KLORRF	R	4	1	VTP	4
KLTAB	R	4	16	FIXDT	64
KLX	R	4	1	VTPDT	4
KLY	R	4	1	VTPDT	4
KLZ	R	4	1	VTPDT	4
KM	I	4	1	FRAME1	4
KMAX	R	4	1	FADCMP	4
KMU	R	4	1	FIXDT	4
KMV	R	4	1	FIXDT	4
KMW	R	4	1	FIXDT	4
KN	I	4	1	FRAME1	4
KPU	R	4	1	FIXDT	4
KPV	R	4	1	FIXDT	4
KPW	R	4	1	FIXDT	4
KRASH	I	2	1	VPFM	2
KS	R	4	1	FRM1	4
KSC	R	4	3	VTPDT	12
KSCX	R	4	1	VTPDT	4
KSCY	R	4	1	VTPDT	4
KSCZ	R	4	1	VTPDT	4
KSF	R	4	1	VTPDT	4
KSKY	R	4	1	VPFM	4
KUVW	R	4	6	FIXDT	24
KV	R	4	1	FRM1	4
KV	R	4	1	VTPDT	4
KW	R	4	1	FRM1	4
LARG	I	4	1	MODSET	4
LAT	R	8	1	LNGLAT	8
LEFT	I	4	1	MODSET	4
LEFT	I	4	1	PUT	4
LEFT	I	4	1	REED	4
LN	I	4	1	MISC	4
LN	R	4	1	MISC	4
LO	I	4	1	MISC	4
LO	R	4	1	MISC	4
LOCFLG	L	4	1	OPTNS	4
LODMOD	L	4	1	OPTNS	4
LONG	R	8	1	LNGLAT	8
LPCT	I	4	1	MODSET	4
LPCT	I	4	1	PUT	4
LPCT	I	4	1	REED	4
LSP	I	4	1	MISC	4
LSP	R	4	1	MISC	4
LST	I	4	1	MISC	4
LST	R	4	1	MISC	4
LTPARM	R	4	2816	TABLS	11264
LUCMN	I	4	1	FRAME1	4
LUREG	I	4	1	FRAME1	4
M	I	4	1	PUT	4
M	I	2	1	VTP	2
MAXRNG	R	4	8	MISC	32
MGKL	R	4	1	VTP	4

Table C-2: Variable List  
for FRAME1  
(Sheet 5 of 10)

SYMBOL	T	S	DIMN	LOCATI	TOTAL
MINRNG	R	4	8	MISC	32
MK	R	4	1	FIXDT	4
MKASG	R	4	1	VTP	4
MKASN	R	4	1	VTP	4
MMAT	R	4	21	MMDAT	84
MMC	R	4	21	MMDAT	84
MMN	R	4	9	FRAME1	36
MMPOS	L	4	1	OPTNS	4
MODEL	I	2	1	FRAME1	2
MSTAR	R	4	9	WNDW	36
N	I	4	1	MODSET	4
N	I	4	1	PUT	4
NA	I	4	1	FRAME1	4
NC	I	4	1	FRAME1	4
NE	I	4	1	FIXDT	4
NE	R	4	1	FIXDT	4
NFSUM	I	4	1	FIXDT	4
NFSUM	R	4	1	FIXDT	4
NFU	R	4	1	VTP	4
NFV	R	4	1	VTP	4
NFW	R	4	1	VTP	4
NFX	R	4	1	VTPDT	4
NFY	R	4	1	VTPDT	4
NFZ	R	4	1	VTPDT	4
NG	R	4	9	WNDW	36
NL	I	4	1	FIXDT	4
NL	R	4	1	FIXDT	4
NLOD	I	4	1	VTPDT	4
NNSEC	I	4	1	FRAME1	4
NOB	I	4	1	FRAME1	4
NOEDB	I	4	1	FR1D	4
NP	R	4	1	VTP	4
NP2	R	4	1	VTP	4
NSEC	I	4	1	FRAME1	4
NSXNVX	R	4	1	VTP	4
NSYNVY	R	4	1	VTP	4
NSZNVZ	R	4	1	VTP	4
NV	R	4	3	VTPDT	12
NVP	I	4	9	FIXDT	36
NVP	R	4	9	FIXDT	36
NVX	R	4	1	VTPDT	4
NVY	R	4	1	VTPDT	4
NVZ	R	4	1	VTPDT	4
OFF	I	4	1	LOD	4
P	R	4	1	VTPDT	4
P1	R	4	1	FADCMP	4
P1	R	4	1	MMFAD	4
P2	R	4	1	FADCMP	4
P2	R	4	1	MMFAD	4
P3	R	4	1	FADCMP	4
P3	R	4	1	MMFAD	4
PF	R	4	1	VTP	4
PH	R	4	1	ROTMAT	4

Table C-2: Variable List  
for FRAME1  
(Sheet 6 of 10)

SYMBOL	T	S	DIMN	LOCATI	TOTAL
PL	R	4	6	VTP	24
PREM	R	4	9	FRAME1	36
PS	R	4	1	ROTMAT	4
PTLFLG	L	4	1	OPTNS	4
R	R	4	1	VTPDT	4
R12NM	R	4	1	FRAME1	4
R2	R	4	1	VTP	4
RADCNV	R	8	1	LNGLAT	8
RADCNV	R	4	1	ROTMAT	4
RADCNV	R	4	1	WINDOW	4
RB	R	4	1	FRM1	4
RB	R	4	1	VTPDT	4
RBF	L	4	1	VTPDT	4
RBOR	R	4	1	VTP	4
RE	I	4	1	FRAME1	4
RE	R	4	1	VTP	4
REGCT	I	4	1	FR1D	4
REGG	R	4	1	VTP	4
RF	R	4	1	VTP	4
RF2	R	4	1	VTP	4
RG	R	4	1	VTP	4
RL	R	4	1	FRM1	4
RMAX	R	4	1	VTP	4
RMID	R	4	1	FRAME1	4
RMRERF	R	4	1	VTP	4
RMRERG	R	4	1	VTP	4
RNG	R	4	1	VTPDT	4
RP	R	4	3	FIXDT	12
RPC	R	4	3	FIXDT	12
RPCX	R	4	1	FIXDT	4
PCY	R	4	1	FIXDT	4
RPCZ	R	4	1	FIXDT	4
RPM	R	4	3	FRAME1	12
RPP	R	4	3	VTPDT	12
RPX	R	4	1	VTPDT	4
RPY	R	4	1	VTPDT	4
RPZ	R	4	1	VTPDT	4
RR	R	4	1	FRM1	4
RRB	R	4	1	VTPDT	4
RS	R	4	1	VTPDT	4
RSLTN	I	4	6	MODSET	24
RSLTN	I	4	6	SETRD	24
RT	R	4	1	FRM1	4
RTST	R	4	1	VTP	4
S1	R	4	1	ROTMAT	4
S2	R	4	1	ROTMAT	4
S3	R	4	1	ROTMAT	4
SG	R	4	1	VTP	4
SIN	R	4	1	LNGLAT	4
SKY	I	2	3	VPFM	6
SN	R	4	3	FRM1	12
SN	R	4	1	VTP	4
SSW	L	4	32	SSWTCH	128

Table C-2: Variable List  
for FRAME1  
(Sheet 7 of 10)

SYMBOL	T	S	DIMN	LOCATI	TOTAL
SV	R	4	3	FIXDT	12
SX	R	4	1	FIXDT	4
SY	R	4	1	FIXDT	4
SZ	R	4	1	FIXDT	4
T	R	8	3	FRAME1	24
T	R	4	1	ROTMAT	4
TBLK	I	4	5	MODSET	20
TBLK	I	4	5	PUT	20
TBLK	I	4	5	REED	20
TBLK	I	4	5	SETRD	20
TEXFLG	L	4	1	OPTNS	4
TM1	R	4	9	FRAME1	36
TM2	R	4	9	FRAME1	36
TMP	R	4	1	VTP	4
TXTAB	I	4	3	MISC	12
UFAD	I	4	16	UNFDT	64
UFC	R	4	3	UNFDT	12
UFDC	R	4	9	UNFDT	36
UFPROC	L	4	1	UNFDT	4
UNITM	R	4	9	LNGLAT	36
UOR	R	4	3	VTPDT	12
UPX	R	4	1	FIXDT	4
UPY	R	4	1	FIXDT	4
UPZ	R	4	1	FIXDT	4
URPRBR	R	4	1	VTP	4
UVSWS	R	4	9	FIXDT	36
UVSWSP	R	4	9	VTPDT	36
UVW	R	4	9	FRM1	36
UX	R	4	1	VTPDT	4
UXXOR	R	4	1	VTP	4
UY	R	4	1	VTPDT	4
UYYOR	R	4	1	VTP	4
UZ	R	4	1	FRM1	4
UZ	R	4	1	VTPDT	4
UZZOR	R	4	1	VTP	4
V	I	2	1	VTP	2
V	R	4	3	VTPDT	12
VF	R	4	1	WINDOW	4
VFOV	R	4	1	FRM1	4
VOR	R	4	1	VTPDT	4
VORNEG	R	4	1	VTP	4
VORPOS	R	4	1	VTP	4
VP	R	8	3	FRM1	24
VPN	R	4	9	FRM1	36
VPX	R	4	1	FIXDT	4
VPY	R	4	1	FIXDT	4
VPZ	R	4	1	FIXDT	4
VX	R	4	1	VTPDT	4
VX1	R	4	1	VTPDT	4
VXXOR	R	4	1	VTP	4
VY	R	4	1	VTPDT	4
VY1	R	4	1	VTPDT	4
VYYOR	R	4	1	VTP	4

Table C-2: Variable List  
for FRAME1  
(Sheet 8 of 10)

SYMBOL	T	S	DIMN	LOCATI	TOTAL
VZ	R	4	1	FRM1	4
VZ	R	4	1	VTPDT	4
VZ1	R	4	1	VTPDT	4
VZZOR	R	4	1	VTP	4
WND	R	4	3	FRM1	12
WNDFLG	L	4	1	JWIN	4
WOR	R	4	1	VTPDT	4
WORNEG	R	4	1	VTP	4
WORPOS	R	4	1	VTP	4
WPX	R	4	1	FIXDT	4
WPY	R	4	1	FIXDT	4
WPZ	R	4	1	FIXDT	4
WR	R	4	1	VTP	4
WS	R	4	1	VTPDT	4
WVP	R	4	9	FRAME1	36
WX	R	4	1	VTPDT	4
WXXOR	R	4	1	VTP	4
WY	R	4	1	VTPDT	4
WYYOR	R	4	1	VTP	4
WZ	R	4	1	FRM1	4
WZ	R	4	1	VTPDT	4
WZZOR	R	4	1	VTP	4
X	R	4	1	VTP	4
XM	R	4	1	FRAME1	4
XOR	R	4	1	VTP	4
XP	R	4	1	VTP	4
XPNFX	R	4	1	VTP	4
XTMP	R	4	20	FIXDT	80
Y	R	4	1	VTP	4
YM	R	4	1	FRAME1	4
YOR	R	4	1	VTP	4
YP	R	4	1	VTP	4
YPNFY	R	4	1	VTP	4
Z	R	8	1	LNGLAT	8
Z	R	4	1	VTP	4
ZC	R	4	1	FADE	4
ZG	R	8	1	FADCMP	8
ZM	R	4	1	FRAME1	4
ZMIN	R	4	1	FADE	4
ZMM	R	4	1	FRAME1	4
ZOR	R	4	1	VTP	4
ZP	R	4	1	VTP	4
ZPNFZ	R	4	1	VTP	4
ZS	R	8	1	FADCMP	8
ZSEA	R	8	1	WNDW	8
ZVP	R	8	1	FRAME1	8

Table C-2: Variable List  
for FRAME1  
(Sheet 9 of 10)

	TOTAL
-----	-----
SIZE	1,906
DIMN	9,812
TOTAL	35,644

Table C-2: Variable List  
for FRAME1  
(Sheet 10 of 10)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
ABSFAN	I	2	4096	F2WRK	8192
ACLIST	I	4	32	F2WRK	128
ACTREG	I	4	128	FRAME2	512
AEXFL	I	4	128	F2WRK	512
AFLG	L	4	1	FACPRO	4
AFN	I	4	1	FACDT	4
AFNO	I	4	1	EDGCAL	4
AK	R	4	1	FIXDT	4
ALPA	R	4	1	FMOD	4
ALPH	R	4	1	FEP	4
ALPHA	R	4	1	FEPDT	4
AMAP	I	4	562	F2WRK	2248
ATT	R	4	3	FRM1	12
AZIM	R	4	1	FRM1	4
BE	R	4	1	CPLITE	4
BETA	R	4	1	FEPDT	4
BL	R	4	1	CPLITE	4
BLND	I	4	1	FACPRO	4
BLNFLG	L	4	1	OPTNS	4
BMS	R	4	1	CPLITE	4
BRDFLG	L	4	1	EDGCAL	4
BUFF	I	4	4096	BFRI	16384
BUFF	I	4	1024	BFRM	4096
BUFF	I	4	4096	BFRO	16384
BUFF	I	4	15	DLCAL	60
BUFF	I	4	5	EDGCAL	20
BUFF	I	4	5	FACOUT	20
BUFF	I	4	15	FACPRO	60
BUFF	I	4	160	FRAME2	640
BUFF	I	4	10	PTCAL	40
BUFFT	I	4	15	FACPRO	60
BUFFV	I	4	5	EDGCAL	20
C	I	2	1	VTP	2
CF1	R	4	1	CPBLND	4
CF2	R	4	1	CPBLND	4
CKA	R	4	1	CPLITE	4
CKB	R	4	1	CPLITE	4
CLD	R	4	1	CPBLND	4
CLR	I	2	3	COL	6
CLRF	I	2	3	COL	6
CLUF	L	4	1	FRAME2	4
CLUST	I	2	1	FRAME2	2
COLOR	R	4	768	TABLS	3072
COLTAB	I	4	10	COLT	40
CTHEP	R	4	1	VTP	4
CTHETA	R	4	1	VTP	4
CV	R	4	1	FRM1	4
CW	R	4	1	FRM1	4
D	R	4	1	FMOD	4
D.AA	I	4	1	COL	4
D.AA	I	4	1	CPBLND	4
D.AA	I	4	1	CPFADE	4
D.AA	I	4	1	CPLITE	4

Table C-3: Variable List  
for FRAME2  
(Sheet 1 of 14)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
D.AA	I	4	1	EDWOUT	4
D.AA	I	4	1	FRAME2	4
D.AA	I	4	1	MULT	4
D.AA	I	4	1	VTP	4
D.AB	I	4	1	CPBLND	4
D.AB	I	4	1	CPFADE	4
D.AB	I	4	1	MULT	4
D.BA	I	4	1	FRAME2	4
D.BA	I	4	1	MULT	4
D.BA	I	4	1	PTCAL	4
D.BB	I	4	1	FRAME2	4
D.BB	I	4	1	MULT	4
D.CA	I	4	1	FRAME2	4
D.CA	I	4	1	MULT	4
D.CA	I	4	1	PTCAL	4
D.DA	I	4	1	FRAME2	4
D.EA	I	4	1	FRAME2	4
D2FLG	L	4	1	FACOUT	4
D3FLG	L	4	1	FACOUT	4
DFP	R	4	1	CPFM	4
DLFLG	L	4	1	FACPRO	4
DUPEDG	L	4	1	FEPDT	4
ECW	R	4	3600	OUT	14400
EDGFLG	L	4	1	OPTNS	4
EDGMOD	L	4	1	FEPDT	4
EDW	I	4	1	EDWOUT	4
EG	I	4	1	EDGCAL	4
EGFLG	L	4	1	FACPRO	4
ELEM1	R	4	1	FEP	4
ELEM2	R	4	1	FEP	4
ELEML	R	4	1	FEP	4
ELEMLX	R	4	1	FEP	4
ELEMR	R	4	1	FEP	4
ELEMRIX	R	4	1	FEP	4
ELEMT1	R	4	1	FEP	4
ELEMT2	R	4	1	FEP	4
ELEV	R	4	1	FRM1	4
ERRMSG	I	4	1	MISC	4
ETOL	R	4	1	FMOD	4
EXF	I	4	1	EDGCAL	4
EXFL	I	4	16	F2WRK	64
EXT	I	4	1	EDGCAL	4
F	R	4	1	CPFADE	4
F	I	2	1	VTP	2
F1RFLG	L	4	1	OPTNS	4
FACE	I	2	1	DLCAL	2
FACE	I	2	1	FACPRO	2
FACFLG	I	4	1	FACPRO	4
FACL	I	4	1	FACDT	4
FACR	I	4	1	FACDT	4
FADFLG	L	4	1	OPTNS	4
FILE	R	8	6	INIT2	48
FILE	R	8	1	UPDATE	8

Table C-3: Variable List  
for FRAME2  
(Sheet 2 of 14)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
FORI	R	4	1	FACPRO	4
FORI	R	4	1	FMOD	4
FORI	R	4	1	MODST2	4
FORI	R	4	1	NEWED	4
FORI	R	4	1	NEWPL	4
FORI	R	4	1	PUT2	4
FORI	R	4	1	REED2	4
FORIP	R	4	1	DLCAL	4
FORITF	R	4	1	FACPRO	4
FORJ	R	4	1	FACPRO	4
FORJ	R	4	1	FMOD	4
FORJ	R	4	1	INIT2	4
FORJ1	R	4	1	MODST2	4
FORJ1	R	4	1	PUT2	4
FORJ1	R	4	1	REED2	4
FORJR	R	4	1	FACOUT	4
FORK	R	4	1	DLCAL	4
FORK	R	4	1	EDGCAL	4
FORK	R	4	1	FACOUT	4
FORM	R	4	1	EDGCAL	4
FORM	R	4	1	PUT2	4
FORNED	R	4	1	EDGCAL	4
FORNF	R	4	1	FACPRO	4
GAMMA	R	4	1	FEP	4
HFOV	R	4	1	FRM1	4
HTOL	R	4	1	FEP	4
I	I	4	1	EDWOUT	4
I	I	4	1	FACPRO	4
I	I	4	1	FMOD	4
I	I	4	1	FRAME2	4
I	I	4	1	MODST2	4
I	I	4	1	MULT	4
I	I	4	1	NEWED	4
I	I	4	1	NEWPL	4
I	I	4	1	PUT2	4
I	I	4	1	REED2	4
I	I	4	1	VTP	4
IO	I	4	1	FRM1	4
I1	I	4	1	FRAME2	4
IABF	I	4	1	F2WRK	4
IABSAD	I	4	1	MODST2	4
IABSAD	I	4	1	SETRD2	4
IAC	I	4	1	F2WRK	4
IACLF	I	4	16	RF	64
IARG	I	4	1	MODST2	4
IARG	I	4	1	PUT2	4
IARG	I	4	1	REED2	4
IARG	I	4	1	SETFIL	4
IARG	I	4	1	SETRD2	4
IB	I	4	1	FACPRO	4
IBFLG	L	4	1	VTPDT	4
IBITWD	I	4	1	NEWED	4
IBLC	I	4	1	FACPRO	4

Table C-3: Variable List  
for FRAME2  
(Sheet 3 of 14)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
IBLK	I	4	1	FACPRO	4
IBLKNO	I	4	1	BLSDT	4
IBLND	I	2	1	COLDT	2
IBLTYP	I	4	1	BLSDT	4
IBMBND	I	4	1	FEP	4
IBNF	I	4	1	FEP	4
IBP	I	4	1	CPBLND	4
IC	I	4	1	FRAME2	4
IC	I	4	3	NEWED	12
IC	I	4	3	NEWPL	12
ICHAN	I	4	1	OPTNS	4
ICHANT	L	4	1	DLCAL	4
ICHASS	L	4	1	VTPDT	4
ICHFLG	L	4	1	VTPDT	4
ICL	I	4	1	FRAME2	4
ICLCT	I	4	1	FRAME2	4
ICLFAD	I	4	1	FRAME2	4
ICOLN	I	4	1	COLT	4
ICOSYS	I	4	1	MISC	4
ICT	I	2	3	CPFADE	6
IDL	I	4	1	FACPRO	4
IDL1	I	4	1	DLCAL	4
IDLAD	I	4	1	DLCAL	4
IDRLAD	I	4	1	RBHDT	4
IDRLD1	I	4	1	DLCAL	4
IDUM	I	2	1	COL	2
IEADUF	I	4	1	UNFDT	4
IECW	I	4	3600	OUT	14400
IEDGAD	I	4	1	RBHDT	4
IEF	I	4	1	MISC	4
IEGSAD	I	4	1	EDGCAL	4
IELAP	I	4	1	FRAME2	4
IEXTF	I	2	1	COLDT	2
IF1	I	2	1	COLDT	2
IF2	I	2	1	COLDT	2
IFACAD	I	4	1	RBHDT	4
IFACPR	I	4	1	FACOUT	4
IFADUF	I	4	1	UNFDT	4
IFATYP	I	4	1	FACOUT	4
IFBN	I	4	1	FRAME2	4
IFF	I	2	1	COLDT	2
IFOGC	I	4	3	CPFM	12
IFXL0D	I	4	1	OPTNS	4
IGNDC	I	4	3	CPFM	12
IHAZC	I	4	3	CPFM	12
IHIB	I	4	1	FEPDT	4
IK	I	4	1	VTP	4
ILBC	R	4	1	FEPDT	4
ILFBND	I	4	1	FEP	4
ILOD	I	4	1	VTPDT	4
IMIN	I	4	1	FRAME2	4
IMODEL	I	4	1	VTPDT	4
IMODNO	I	4	1	FACOUT	4

Table C-3: Variable List  
for FRAME2  
(Sheet 4 of 14)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
IN2FC	I	4	1	FEP	4
INCT	I	4	6	VTPDT	24
INF	I	4	1	FACPRO	4
INFV	I	4	1	FEP	4
INFV1	I	4	1	FEP	4
INFV1P	I	4	1	FEP	4
INFV2	I	4	1	FEP	4
INFV2P	I	4	1	FEP	4
INOFED	I	4	1	FACPRO	4
INXTBN	I	4	1	BLSDT	4
IOBJNO	I	4	1	FACOUT	4
IOS	I	2	1	COLDT	2
IP	I	4	1	DLCAL	4
IP	I	4	1	FRAME2	4
IPLCW	I	4	800	OUT	3200
IPROC	I	4	1	MODST2	4
IPROC	I	4	1	PUT2	4
IPROC	I	4	1	REED2	4
IPROC	I	4	1	SETRD2	4
IPTLAD	I	4	1	RBHDT	4
IPTLT	I	2	1	COLDT	2
IR	I	4	1	BLSDT	4
IRBC	R	4	1	FEPDT	4
IRCLN	I	4	1	CLUSPR	4
IREC	I	4	8	BFRI	32
IREC	I	4	8	BFRM	32
IREC	I	4	8	BFRO	32
IREL	I	4	1	BLSDT	4
IREFN	I	4	1	FACOUT	4
IREP1	I	4	1	FEP	4
IREP12	I	4	1	FEP	4
IREP2	I	4	1	FEP	4
IREPL	I	4	1	FEP	4
IREPR	I	4	1	FEP	4
IRFC	I	4	1	MODST2	4
IRFC	I	4	1	PUT2	4
IRFC	I	4	1	REED2	4
IRFC	I	4	1	SETRD2	4
IRFNO	I	4	1	FACPRO	4
IRN	I	4	1	RBHDT	4
IRPLAD	I	4	1	PTCAL	4
IRSLTN	I	4	7	NDXTBS	28
IRTBND	I	4	1	FEP	4
IRX	I	4	1	MODST2	4
IRX	I	4	1	SETRD2	4
ISEC	I	4	1	FRAME2	4
ISFPTR	I	4	1	CLUSPR	4
ISKYC	I	4	3	CPFM	12
ISPF	I	4	1	VTP	4
ISPL	I	4	1	VTP	4
ISTAT	I	4	1	FRAME2	4
ISTAT	I	4	1	INIT2	4
ISTAT	I	4	1	MODST2	4

Table C-3: Variable List  
for FRAME2  
(Sheet 5 of 14)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
ISTAT	I	4	1	PUT2	4
ISTAT	I	4	1	REED2	4
ISTAT	I	4	1	SETFIL	4
ISTAT	I	4	1	SETRD2	4
ISTAT	I	4	1	UPDATE	4
ISTIM	I	4	3	FRAME2	12
ITADUF	I	4	1	UNFDT	4
ITAFIL	I	4	1	F2WRK	4
ITEX	I	4	1	MODT	4
ITF	I	4	1	F2WRK	4
ITFR	I	4	1	FACOUT	4
ITFST	I	4	1	FACPRO	4
ITIM	I	4	3	FRAME2	12
ITNAF	I	4	1	F2SUM	4
ITNED	I	4	1	F2SUM	4
ITPBND	I	4	1	FEP	4
ITRN	I	4	1	EDGCAL	4
ITXF	I	4	1	FMOD	4
ITXRAD	I	4	1	FACPRO	4
ITXS	I	4	16	MODT	64
ITXTAD	I	4	1	RBHDT	4
IU	I	4	1	UNFDT	4
IUAD	I	4	1	FRAME2	4
IUF	I	4	1	FRAME2	4
IUFRAD	I	4	1	FRAME2	4
IUNFAD	I	4	1	RBHDT	4
IUP	I	4	1	MODST2	4
IUP	I	4	1	PUT2	4
IUP	I	4	1	REED2	4
IVRTAD	I	4	1	EDGCAL	4
IW	I	4	1	FACPRO	4
IX	I	4	8	BFRI	32
IX	I	4	8	BFRM	32
IX	I	4	8	BFRO	32
IXCOL	I	4	1	FACDT	4
IXX	I	4	1	FRAME2	4
J	I	4	1	DLCAL	4
J	I	4	1	FACPRO	4
J	I	4	1	FMOD	4
J	I	4	1	FRAME2	4
J	I	4	1	INIT2	4
J	I	4	1	MULT	4
JO	I	4	1	FRM1	4
J1	I	4	1	EDWOUT	4
J1	R	4	1	FEP	4
J1	I	4	1	FRAME2	4
J2	I	4	1	EDWOUT	4
J2	R	4	1	FEP	4
J3	I	4	1	EDWOUT	4
J4	I	4	1	EDWOUT	4
J5	I	4	1	EDWOUT	4
JARG	I	4	1	MODST2	4
JARG	I	4	1	PUT2	4

Table C-3: Variable List  
for FRAME2  
(Sheet 6 of 14)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
JARG	I	4	1	REED2	4
JARG	I	4	1	SETRD2	4
JBLK	I	4	1	FACOUT	4
JLBC	R	4	1	FEPDT	4
JMA	I	4	1	FRAME2	4
JP	I	4	1	FRAME2	4
JPROC	I	4	1	MODST2	4
JR	I	4	1	FACOUT	4
JRA	I	4	1	FRAME2	4
JR3C	R	4	1	FEPDT	4
JREC	I	4	1	REED2	4
JRFC	I	4	1	MODST2	4
JSSW	I	4	1	MISC	4
JX	I	4	1	MODST2	4
JX	I	4	1	PUT2	4
JX	I	4	1	REED2	4
K	I	4	1	COL	4
K	I	4	1	CPBLND	4
K	I	4	1	CPFADE	4
K	I	4	1	CPLITE	4
K	I	4	1	DLCAL	4
K	I	4	1	EDGCAL	4
K	I	4	1	FACOUT	4
K	I	4	1	FACPRO	4
K	I	4	1	FRAME2	4
K	I	4	1	MODST2	4
K	I	4	1	MULT	4
K	I	4	1	PTCAL	4
K	I	4	1	PUT2	4
K	I	4	1	REED2	4
KARG	I	4	1	MODST2	4
KARG	I	4	1	PUT2	4
KEDG	I	4	1	OUT	4
KEDGT	I	4	1	OUT	4
KFAC	I	4	1	EDGCAL	4
KI	R	4	1	FRM1	4
KIJ	R	4	4	FIXDT	16
KIMIO	R	4	1	FIXDT	4
KINP	R	4	12	MODT	48
KIPIO	R	4	1	FIXDT	4
KJ	R	4	1	FRM1	4
KJMJO	R	4	1	FIXDT	4
KJPJO	R	4	1	FIXDT	4
KL	R	4	3	VTPDT	12
KLDF	R	4	1	VTPDT	4
KLDREZ	R	4	1	VTP	4
KLE	R	4	1	VTPDT	4
KLIT	I	4	1	OUT	4
KLITT	I	4	1	OUT	4
KLM	R	4	1	VTPDT	4
KLMREP	R	4	1	VTP	4
KLOD	I	4	1	FIXDT	4
KLOR	R	4	1	VTP	4

Table C-3: Variable List  
for FRAME2  
(Sheet 7 of 14)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
KLORRF	R	4	1	VTP	4
KLTAB	R	4	16	FIXDT	64
KLX	R	4	1	VTPDT	4
KLY	R	4	1	VTPDT	4
KLZ	R	4	1	VTPDT	4
KMU	R	4	1	FIXDT	4
KMV	R	4	1	FIXDT	4
KMW	R	4	1	FIXDT	4
KNPN	R	4	3	FMOD	12
KNPU	R	4	3	FMOD	12
KP	I	4	1	FRAME2	4
KPU	R	4	1	FIXDT	4
KPV	R	4	1	FIXDT	4
KPW	R	4	1	FIXDT	4
KS	R	4	1	FRM1	4
KSC	R	4	3	VTPDT	12
KSCX	R	4	1	VTPDT	4
KSCY	R	4	1	VTPDT	4
KSCZ	R	4	1	VTPDT	4
KSF	R	4	1	VTPDT	4
KUVW	R	4	6	FIXDT	24
KV	R	4	1	VTPDT	4
LAST	I	4	3	FACOUT	12
LCP	I	4	1	CPLITE	4
LEFT	I	4	1	MODST2	4
LEFT	I	4	1	PUT2	4
LEFT	I	4	1	REED2	4
LF	I	2	1	COLDT	2
LFFLG	L	4	1	FACPRO	4
LI1	R	4	1	FEP	4
LI2	R	4	1	FEP	4
LINE1	R	4	1	FEP	4
LINE2	R	4	1	FEP	4
LINEL	R	4	1	FEP	4
LINELX	R	4	1	FEP	4
LINER	R	4	1	FEP	4
LINERX	R	4	1	FEP	4
LITTOT	I	4	1	PTCAL	4
LN	I	4	1	MISC	4
LO	I	4	1	MISC	4
LOCFLG	L	4	1	OPTNS	4
LODMOD	L	4	1	OPTNS	4
LP	I	4	1	EDWOUT	4
LPB	L	4	1	FACPRO	4
LPCT	I	4	1	MODST2	4
LPCT	I	4	1	PUT2	4
LPCT	I	4	1	REED2	4
LPFLG	L	4	1	FACPRO	4
LRGBNO	I	4	1	BLSDT	4
LSP	I	4	1	MISC	4
LST	I	4	1	MISC	4
LTMOD	I	4	1	COL	4
LTNENF	I	4	1	PTCAL	4

Table C-3: Variable List  
for FRAME2  
(Sheet 8 of 14)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
LTNINH	I	4	1	PTCAL	4
LTOTEF	I	4	1	PTCAL	4
LTOTIH	I	4	1	PTCAL	4
LTPARM	R	4	2816	TABLS	11264
LUN	I	4	1	EDGCAL	4
LUT	I	4	8	NDXTBS	32
M	I	4	1	EDGCAL	4
M	I	4	1	PUT2	4
M	I	2	1	VTP	2
MAXRNG	R	4	8	MISC	32
MAXSIZ	R	4	1	CPLITE	4
MCM	I	2	3	COLDT	6
MFN	I	4	1	FRAME2	4
MGKL	R	4	1	VTP	4
MINRNG	R	4	8	MISC	32
MINSIZ	R	4	1	CPLITE	4
MINSZF	I	2	1	COLDT	2
MINUS	I	4	1	EDWOUT	4
MK	R	4	1	FIXDT	4
MKASG	R	4	1	VTP	4
MKASN	R	4	1	VTP	4
MMPOS	L	4	1	OPTNS	4
MN	I	4	1	NEWED	4
MN	I	4	1	NEWPL	4
MNE	I	4	1	EDGCAL	4
MNH	I	4	1	NEWED	4
MO	I	4	1	NEWED	4
MO	I	4	1	NEWPL	4
MODEL	I	2	1	FRAME2	2
MODF	L	4	1	FACPRO	4
MODRT	L	4	1	NEWED	4
MOH	I	4	1	NEWED	4
MSKE	I	4	1	EDWOUT	4
MSKL	I	4	1	EDWOUT	4
MXEDG	I	4	1	NEWED	4
MXLIT	I	4	1	NEWPL	4
N	I	4	1	CPLITE	4
N	I	4	1	MODST	4
N	I	4	1	NEWED	4
N	I	4	1	NEWPL	4
N	I	4	1	PUT2	4
NAA	I	4	1	NEWED	4
NAFLG	L	4	1	FACPRO	4
NAMAP	I	4	562	F2WRK	2248
NE	I	4	1	FIXDT	4
NE2	R	4	1	FEP	4
NED	I	4	1	EDGCAL	4
NEDG	I	4	1	EDGCAL	4
NF	I	4	1	FACPRO	4
NF	I	4	1	NEWED	4
NF	I	4	1	NEWPL	4
NF	R	4	3	VTPDT	12
NF1	I	4	1	FACDT	4

Table C-3: Variable List  
for FRAME2  
(Sheet 9 of 14)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
NFF	R	4	3	FMOD	12
NFG	R	4	3	FMOD	12
NFSUM	I	4	1	FIXDT	4
NFTOT	I	4	1	FACDT	4
NFU	R	4	1	VTP	4
NFV	R	4	1	VTP	4
NFW	R	4	1	VTP	4
NFX	R	4	1	VTPDT	4
NFY	R	4	1	VTPDT	4
NFZ	R	4	1	VTPDT	4
NHDR	I	4	1	EDWOUT	4
NL	I	4	1	FIXDT	4
NLEL	R	4	1	COLDT	4
NLIT	I	4	1	PTCAL	4
NLOD	I	4	1	VTPDT	4
NNSEC	I	4	1	FRAME2	4
NOCLFA	I	4	1	FRAME2	4
NOSR	I	4	1	FRAME2	4
NP	I	4	1	PTCAL	4
NP	R	4	1	VTP	4
NP2	R	4	1	VTP	4
NPL	I	4	1	PTCAL	4
NPLS	I	4	1	PTCAL	4
NSEC	I	4	1	FRAME2	4
NSF1	I	4	1	FRAME2	4
NSXNVX	R	4	1	VTP	4
NSYNVY	R	4	1	VTP	4
NSZNVZ	R	4	1	VTP	4
NTFF	I	4	1	FACDT	4
NUF	I	4	1	FRAME2	4
NV	R	4	3	VTPDT	12
NVP	I	4	9	FIXDT	36
NVX	R	4	1	VTPDT	4
NVY	R	4	1	VTPDT	4
NVZ	R	4	1	VTPDT	4
NXF	I	4	1	FACPRO	4
NXTFST	I	4	1	FACPRO	4
P	R	4	1	VTPDT	4
PO	R	4	1	FMOD	4
PF	R	4	1	VTP	4
PL	R	4	6	VTP	24
PLCW	R	4	800	OUT	3200
PRTAB	R	4	48	CLUSPR	192
PSEU	R	4	2	FEPDT	8
PTLFLG	L	4	1	OPTNS	4
R	R	4	1	CPFADE	4
R	R	4	1	VTPDT	4
R2	R	4	1	COL	4
R2	R	4	1	VTP	4
RA2	R	4	1	CPLITE	4
RB	R	4	1	FRM1	4
RB	R	4	1	VTPDT	4
RBF	L	4	1	VTPDT	4

Table C-3: Variable List  
for FRAME2  
(Sheet 10 of 14)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
RBOR	R	4	1	VTP	4
RBUFF	R	4	5	FACOUT	20
RCONST	R	4	1	CPFADE	4
RDUM	R	4	1	COL	4
RE	R	4	1	VTP	4
RE2	R	4	1	CPLITE	4
RERG	R	4	1	VTP	4
RF	R	4	1	VTP	4
RF2	R	4	1	VTP	4
RG	R	4	1	VTP	4
RI	I	4	1	EDGCAL	4
RJ	I	4	1	EDGCAL	4
RL	R	4	1	FRM1	4
RMAX	R	4	1	FEP	4
RMAX	R	4	1	VTP	4
RMRERF	R	4	1	VTP	4
RMRERG	R	4	1	VTP	4
RNG	R	4	1	CPLITE	4
RNG	R	4	1	VTPDT	4
RP	R	4	3	FIXDT	12
RP1	R	4	1	FRAME2	4
RPC	R	4	3	FIXDT	12
RPCX	R	4	1	FIXDT	4
PCY	R	4	1	FIXDT	4
RPCZ	R	4	1	FIXDT	4
RPF	L	4	1	CLUSPR	4
RPP	R	4	3	VTPDT	12
RPX	R	4	1	VTPDT	4
RPY	R	4	1	VTPDT	4
RPZ	R	4	1	VTPDT	4
RR	R	4	1	FRM1	4
RRB	R	4	1	VTPDT	4
RS	R	4	1	VTPDT	4
RT	R	4	1	FRM1	4
RTAFL	R	4	2944	F2WRK	11776
RTST	R	4	1	VTP	4
SG	R	4	1	VTP	4
SLOPE	R	4	1	FEPDT	4
SN	R	4	3	FRM1	12
SN	R	4	1	VTP	4
SNFLG	L	4	1	FACPRO	4
SPFLG	L	4	1	FACOUT	4
SSW	L	4	32	SSWTCH	128
STORE	R	4	1	FEP	4
SV	R	4	3	FIXDT	12
SX	R	4	1	FIXDT	4
SY	R	4	1	FIXDT	4
SZ	R	4	1	FIXDT	4
T16	R	4	1	CPLITE	4
TAFLLST	I	4	2944	F2WRK	11776
TBLK	I	4	5	MODST2	20
TBLK	I	4	5	PUT2	20
TBLK	I	4	5	REED2	20

Table C-3: Variable List  
for FRAME2  
(Sheet 11 of 14)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
TBLK	I	4	5	SETRD2	20
TEXFLG	L	4	1	OPTNS	4
TFFLG	L	4	1	FACOUT	4
TFFLG	L	4	1	FACPRO	4
TFFLG	L	4	1	NEWED	4
TFR	R	4	1	FACDT	4
TFRTMP	R	4	1	FACOUT	4
TMP	R	4	1	VTP	4
TOL	R	4	1	EDGCAL	4
TOL	R	4	1	FEP	4
TOLUP	R	4	1	FEP	4
TXS	R	4	16	MODT	64
TXTAB	I	4	3	MISC	12
U	R	4	1	FRAME2	4
U1R	R	4	1	FEP	4
U2R	R	4	1	FEP	4
UF1	I	4	1	FRAME2	4
UFAD	I	4	16	UNFDT	64
UFC	R	4	3	UNFDT	12
UFDC	R	4	9	UNFDT	36
UFFLG	L	4	1	FRAME2	4
UFPROC	L	4	1	UNFDT	4
UM	R	4	48	FEPDT	192
UNSHIL	R	4	1	FEPDT	4
UNSHIR	R	4	1	FEPDT	4
UNSHJL	R	4	1	FEPDT	4
UNSHJR	R	4	1	FEPDT	4
UOR	R	4	3	VTPDT	12
UPX	R	4	1	FIXDT	4
UPY	R	4	1	FIXDT	4
UPZ	R	4	1	FIXDT	4
URPRBR	R	4	1	VTP	4
URTAB	R	4	48	FRAME2	192
UVSWS	R	4	9	FIXDT	36
UVSWSP	R	4	9	VTPDT	36
UVW	R	4	9	FRM1	36
UX	R	4	1	VTPDT	4
UXXOR	R	4	1	VTP	4
UY	R	4	1	VTPDT	4
UYTOR	R	4	1	VTP	4
UZ	R	4	1	VTPDT	4
UZZOR	R	4	1	VTP	4
V	I	2	1	VTP	2
V	R	4	3	VTPDT	12
V1	R	4	3	FRAME2	12
V1P	R	4	1	FEP	4
V1R	R	4	1	FEP	4
V2P	R	4	1	FEP	4
V2R	R	4	1	FEP	4
VERT	I	2	1	EDGCAL	2
VERT	I	2	1	PTCAL	2
VFOV	R	4	1	FRM1	4
VOR	R	4	1	VTPDT	4

Table C-3: Variable List  
for FRAME2  
(Sheet 12 of 14)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
VORNEG	R	4	1	VTP	4
VORPOS	R	4	1	VTP	4
VP	R	8	3	FRM1	24
VPN	R	4	9	FRM1	36
VPX	R	4	1	FIXDT	4
VPY	R	4	1	FIXDT	4
VPZ	R	4	1	FIXDT	4
VR	R	4	3	FMOD	12
VX	R	4	1	VTPDT	4
VX1	R	4	1	VTPDT	4
VXXOR	R	4	1	VTP	4
VY	R	4	1	VTPDT	4
VY1	R	4	1	VTPDT	4
VYYOR	R	4	1	VTP	4
VZ	R	4	1	VTPDT	4
VZ1	R	4	1	VTPDT	4
VZZOR	R	4	1	VTP	4
W1P	R	4	1	FEP	4
W1R	R	4	1	FEP	4
W2P	R	4	1	FEP	4
W2R	R	4	1	FEP	4
WND	R	4	3	FRM1	12
WOR	R	4	1	VTPDT	4
WORNEG	R	4	1	VTP	4
WORPOS	R	4	1	VTP	4
WPX	R	4	1	FIXDT	4
WPY	R	4	1	FIXDT	4
WPZ	R	4	1	FIXDT	4
WR	R	4	1	VTP	4
WS	R	4	1	VTPDT	4
WX	R	4	1	VTPDT	4
WXXOR	R	4	1	VTP	4
WY	R	4	1	VTPDT	4
WYYOR	R	4	1	VTP	4
WZ	R	4	1	VTPDT	4
WZZOR	R	4	1	VTP	4
X	R	4	1	VTP	4
XOR	R	4	1	VTP	4
XP	R	4	1	VTP	4
XPNFX	R	4	1	VTP	4
Y	R	4	1	VTP	4
YOR	R	4	1	VTP	4
YP	R	4	1	VTP	4
YPNFX	R	4	1	VTP	4
Z	R	4	1	VTP	4
ZOR	R	4	1	VTP	4
ZP	R	4	1	VTP	4
ZPNFZ	R	4	1	VTP	4

Table C-3: Variable List  
for FRAME2  
(Sheet 13 of 14)

	TOTAL
-----	-----
SIZE	2,698
DIMN	34,398
TOTAL	129,376

Table C-3: Variable List  
for FRAME2  
(Sheet 14 of 14)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
A	I	2	1	DECODE	2
A	I	4	1	PRFMOD	4
A	I	4	1	TSBNST	4
A	I	4	1	TSBSNO	4
A	I	4	1	TSEMOV	4
AOE	R	4	1	PREDGA	4
AOM	R	4	18	PRESM	72
AOU	R	4	4	PRLD	16
AOUK	R	4	1	PRAUPD	4
A1	R	4	1	VPAREA	4
A1A	R	4	90	PRVP	360
A1B	R	4	90	PRVP	360
A1E	R	4	1	PREDGA	4
A1F	R	4	1	VPAREA	4
A1KFC	R	4	1	VPCFC	4
A1M	R	4	18	PRESM	72
A1U	R	4	4	PRLD	16
A1UK	R	4	1	PRAUPD	4
A2	R	4	1	VPAREA	4
A2A	R	4	90	PRVP	360
A2B	R	4	90	PRVP	360
A2F	R	4	1	VPAREA	4
A2KFC	R	4	1	VPCFC	4
A2OBJ	R	4	2	PROUT1	8
A3	R	4	1	VPAREA	4
A3F	R	4	1	VPAREA	4
A3KFC	R	4	1	VPCFC	4
AA	R	4	1	VPAREA	4
AB	R	4	1	VPAREA	4
ABSFAC	I	2	256	EDGORD	512
ABSFAN	I	2	4096	GEN	8192
ABSPRI	R	4	256	EDREL	1024
ABVBK	I	2	1	PRVIS	2
AC	R	4	1	VPAREA	4
AE	R	4	12	TSTXEV	48
AK	R	4	1	FIXDT	4
AL	R	4	1	VIDPRO	4
ALF	R	4	40	PLGVP	160
ALFMAX	R	4	1	TSLODS	4
ALFTL	R	4	2	LOCAL	8
ALIM	R	4	90	PRVP	360
ALL	R	4	40	PLGVP	160
ALLTL	R	4	2	LOCAL	8
ALM	R	4	40	PLGVP	160
ALMTL	R	4	2	LOCAL	8
ALPHA	R	4	6	TSLODV	24
AM	I	2	1	PRFBKU	2
AM	I	4	1	PRFMOD	4
AOBJL	I	2	1	FLGS3	2
AOVERB	I	2	1	PROUT2	2
AP1	I	2	1	FLGS4	2
AP1BK	I	2	1	PRVIS	2
AP1NE	I	2	1	PRFONM	2

Table C-4: Variable List  
for FRAME3  
(Sheet 1 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
AP1P	I	2	1	FLGS4	2
AP1P2V	I	2	1	PRFONM	2
APB	I	2	256	PRAPL	512
APBG	I	2	1	PREDGA	2
APT	I	2	256	PRAPL	512
AREAO	R	4	2	PROUT1	8
ARMAX	R	4	1	PRAREA	4
ATEMP	R	4	1	TSEA	4
ATT	R	4	3	FRM1	12
AZIM	R	4	1	FRM1	4
B	I	2	1	DECODE	2
B	I	4	1	PRFMOD	4
B	I	4	1	TSBNST	4
B	I	4	1	TSBSNO	4
B	I	4	1	TSEMOV	4
BB	I	2	3	PRLD	6
BBFO	I	2	1	PRFONM	2
BBM	I	2	18	PRESM	36
BBTM	I	2	1	PRFMOD	2
BCK	I	2	1	PROUT2	2
BCKOFF	R	4	1	PARSEL	4
BCOL	I	4	6	RAMCOM	24
BGCNT	I	2	1	PRCTRL	2
BGNDFL	I	2	1	PROUT1	2
BGNDSL	I	2	1	PROUT1	2
BGRTFL	I	2	1	MODOUT	2
BIGNO	R	4	1	PRAREA	4
BKGB	I	2	1	PREDGA	2
BKGC3	I	2	1	VPMLF	2
BKGC3	I	2	1	VPSIMP	2
BKGL	I	2	1	PREDGE	2
BKGLTP	I	2	8	PRTPL	16
BKGR	I	2	1	PREDGE	2
BKGRTP	I	2	8	PRTPL	16
BKGT	I	2	1	PREDGA	2
BL	I	2	4	PRLD	8
BLIM	R	4	90	PRVP	360
BLM	I	2	18	PRESM	36
BLNFLG	L	4	1	OPTNS	4
BLOCK	C	1	1	WORK4	1
BM	I	2	1	PRFBKU	2
BM	I	4	1	PRFMOD	4
BNX	I	2	1	PROUT1	2
BNX	I	2	3	TSPD1	6
BNXN	I	2	1	TSPD1N	2
BNXP	I	2	1	TSPD1	2
BP1BK	I	2	1	PRVIS	2
BR	I	2	4	PRLD	8
BRM	I	2	18	PRESM	36
BT	I	2	3	PRLD	6
BTFO	I	2	1	PRFONM	2
BTM	I	2	18	PRESM	36
BTOP	I	2	1	PRFMOD	2

Table C-4: Variable List  
for FRAME3  
(Sheet 2 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
BUFF	I	4	320	BFRI	1280
BUFF	I	4	320	BFRM	1280
BUFF	I	4	320	BFRO	1280
C1	I	2	3	VPCFC	6
C2	R	4	6	TSPD2	24
C2	I	2	3	VPCFC	6
C2N	R	4	3	TSPD2N	12
C3	R	4	6	TSPD2	24
C3	I	2	3	VPCFC	6
C3N	R	4	3	TSPD2N	12
C5	R	4	2	TSPD2	8
C5N	R	4	1	TSPD2N	4
C6	R	4	2	TSPD2	8
C6N	R	4	1	TSPD2N	4
CA	I	2	270	PRVP	540
CASE	I	2	5	LR2	10
CASE	I	2	18	TB2	36
CASELE	I	2	15	SINGS	30
CASENO	I	2	8	SINGS	16
CB	I	2	270	PRVP	540
CBKG	I	2	3	VPCFC	6
CC	I	2	270	PRVP	540
CDN	R	4	1	TSPD1N	4
CDSP1	R	4	4	TSDBN	16
CDSUM	R	4	4	TSINC	16
CE	I	2	3	VIDPRO	6
CE	I	2	3	VPSIMP	6
CFBB	I	4	1	FBTB	4
CFBT	I	4	1	FBTB	4
CHAN	I	2	1	TSPD1	2
CHANN	I	2	1	TSPD1N	2
CHAZ	I	2	3	VPCFC	6
CHNSTA	I	2	1	PRFBKD	2
CL	I	2	3	VIDPRO	6
CLLI	R	4	1	VPLLI	4
CLRCH2	I	2	8	RAMSET	16
CLRCH4	I	4	4	RAMSET	16
CLT	I	2	120	PLGVP	240
CLUCEN	R	4	192	WORK4	768
CLUMAP	I	2	18	WORK4	36
CNN	R	4	3	TSPD1N	12
CNSOLE	I	4	1	RAMOUT	4
CNSOLE	I	4	1	RAMSET	4
CNSP1	R	4	12	TSDBN	48
CNSUM	R	4	12	TSINC	48
CO	I	2	90	PRVP	180
COL	I	4	200	GEN	800
COLFLG	I	2	1	ARIN	2
COLOR2	I	2	260	RAMSET	520
COLOR3	I	2	1	ARIN	2
COLOR4	I	4	130	RAMSET	520
COM	R	8	1	INIT3	8
CORCX	I	4	1	WORK4	4

Table C-4: Variable List  
for FRAME3  
(Sheet 3 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
CORCY	I	4	1	WORK4	4
CORCZ	I	4	1	WORK4	4
CORLAT	I	4	1	WORK4	4
CORLON	I	4	1	WORK4	4
CS	I	2	90	PRVP	180
CSI	I	2	1	WORK4	2
CTLWD	I	4	1	WORK11	4
CTLWRD	I	4	20	WORK15	80
CUMMUL	I	2	1	WORK4	2
CURMAP	I	4	40	WORK6	160
CV	R	4	1	FRM1	4
CW	R	4	1	FRM1	4
D.AA	I	4	1	CXMAP	4
D.AA	I	4	1	DECODE	4
D.AA	I	4	1	EDGGEN	4
D.AA	I	4	1	FRAME3	4
D.AA	I	4	1	MODSET	4
D.AA	I	4	1	ORDER	4
D.AA	I	4	1	OVERID	4
D.AA	I	4	1	PRAUPD	4
D.AA	I	4	1	PRCLR	4
D.AA	I	4	1	PRELOD	4
D.AA	I	4	1	PRINIT	4
D.AA	I	4	1	PRTPLU	4
D.AA	I	4	1	PTLSIT	4
D.AA	I	4	1	PUT	4
D.AA	I	4	1	RAMOUT	4
D.AA	I	4	1	RAMSET	4
D.AA	I	4	1	STPED	4
D.AA	I	4	1	STPLT	4
D.AA	I	4	1	TSDBN	4
D.AA	I	4	1	TSEA	4
D.AA	I	4	1	TSEDA	4
D.AA	I	4	1	TSEDGR	4
D.AA	I	4	1	TSEMOV	4
D.AA	I	4	1	TSINIT	4
D.AA	I	4	1	TSLODS	4
D.AA	I	4	1	TSMUX	4
D.AA	I	4	1	TSPINC	4
D.AA	I	4	1	TSSHAD	4
D.AA	I	4	1	TSTXMD	4
D.AA	I	4	1	VPAINC	4
D.AA	I	4	1	VPCFC	4
D.AA	I	4	1	VPFADE	4
D.AA	I	4	1	VPLTC	4
D.AA	I	4	1	VPMLF	4
D.AB	I	4	1	CXMAP	4
D.AB	I	4	1	ORDER	4
D.AB	I	4	1	PRAUPD	4
D.AB	I	4	1	PRELOD	4
D.AB	I	4	1	PRINIT	4
D.AB	I	4	1	PRTPLU	4
D.AB	I	4	1	TSDBN	4

Table C-4: Variable List  
for FRAME3  
(Sheet 4 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
D.AB	I	4	1	TSEDA	4
D.AB	I	4	1	TSEDGR	4
D.AB	I	4	1	TSEMOV	4
D.AB	I	4	1	TSINIT	4
D.AB	I	4	1	TSLODS	4
D.AB	I	4	1	TSMUX	4
D.AB	I	4	1	TSPINC	4
D.AB	I	4	1	TSSHAD	4
D.AB	I	4	1	TSTXMD	4
D.AB	I	4	1	VPAINC	4
D.AB	I	4	1	VPLTC	4
D.AB	I	4	1	VPMLF	4
D.AC	I	4	1	VPAINC	4
D.BA	I	4	1	CXMAP	4
D.BA	I	4	1	ORDER	4
D.BA	I	4	1	STPED	4
D.BA	I	4	1	TSBSNO	4
D.BA	I	4	1	TSINIT	4
D.BA	I	4	1	TSLODS	4
D.BA	I	4	1	TSMUX	4
D.BA	I	4	1	TSPINC	4
D.BA	I	4	1	TSSHAD	4
D.BA	I	4	1	TSTXMD	4
D.BA	I	4	1	VIDPRO	4
D.BA	I	4	1	VPSIMP	4
D.BB	I	4	1	ORDER	4
D.BB	I	4	1	STPED	4
D.BB	I	4	1	TSMUX	4
D.BB	I	4	1	TSPINC	4
D.BB	I	4	1	VPSIMP	4
D.CA	I	4	1	TSINIT	4
DA	R	4	1	PRAUPD	4
DA	R	4	1	TSVPFD	4
DADJ	R	4	4	PRLD	16
DADJA	R	4	90	PRVP	360
DADJB	R	4	90	PRVP	360
DADJE	R	4	1	PREDGA	4
DADJM	R	4	18	PRESM	72
DADJO	R	4	2	PROUT1	8
DB	R	4	1	TSVPFD	4
DDA	R	4	1	TSVPFD	4
DDB	R	4	1	TSVPFD	4
DELI	R	4	1	EDGGEN	4
DELTJ	R	4	1	PATPRO	4
DELTI	R	4	1	PATPRO	4
DEVICE	I	2	1	RAMSET	2
DF343	R	4	1	TSVPFD	4
DHDR	I	4	1	FRM3	4
DI	R	4	1	TSEDA	4
DIDJ	R	4	1	TSEA	4
DIL	R	4	1	TSEDA	4
DIR	R	4	1	TSEDA	4
DJ	R	4	1	TSEDA	4

Table C-4: Variable List  
for FRAME3  
(Sheet 5 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
DJ1	R	4	1	TSEA	4
DJ2	R	4	1	TSEA	4
DJBOT	R	4	1	TSEDA	4
DJMAX	R	4	1	TSEA	4
DJMIN	R	4	1	TSEA	4
DJN	R	4	1	TSBSNO	4
DJTOP	R	4	1	TSEDA	4
DOUT	R	4	1	TSLOD	4
DP	R	4	12	TSLODV	48
DP1	R	4	1	TSLODS	4
DP2	R	4	1	TSLODS	4
DPJ	R	4	6	TSBND	24
DPJM1	R	4	6	TSBND	24
DPJP1	R	4	6	TSBND	24
DPJX	R	4	1	TSDBN	4
DPL	R	4	3	TSBND	12
DPLM1	R	4	3	TSBND	12
DPLP1	R	4	3	TSBND	12
DPMAX	R	4	1	TSLODS	4
DPSUM	R	4	2	TSOUT	8
DRA	R	4	1	VPFAD	4
DRB	R	4	1	VPFAD	4
EC	I	2	12	DECODE	24
ECASE	I	2	3	TSBNSF	6
ECWL	I	4	1	CXMAP	4
ECWL	I	4	1	STPED	4
ECWL	I	4	1	STPLT	4
EDGENO	I	4	1	NSEGE	4
EDGENO	I	4	1	PREDGR	4
EDGENO	I	4	1	PROUT	4
EDGFLG	L	4	1	OPTNS	4
EDW	R	8	1	INIT3	8
EEA	I	2	1	PRFMOD	2
EEB	I	2	1	PRFMOD	2
EEM	I	2	18	PRESM	36
EGF	I	4	1	TSTXMD	4
EHDR	I	4	1	STPED	4
EJ	I	2	4	PRLD	8
EJE	I	2	1	PREDGA	2
EJL	R	4	1	PREDGE	4
EJLEFT	I	2	1	NSEGE	2
EJM	I	2	18	PRESM	36
EJR	R	4	1	PREDGE	4
EJ RTP	R	4	8	PRTPL	32
ELEV	R	4	1	FRM1	4
ENVBLK	I	2	1	WORK4	2
EOF	I	4	1	REED	4
EPRA	I	2	1	PREDGA	2
EPRAB	I	2	1	PREDGA	2
EPRAT	I	2	1	PREDGA	2
ERRMSG	I	4	1	MISC	4
ESMB	I	2	8	PRESMI	16
F1RFLG	L	4	1	OPTNS	4

Table C-4: Variable List  
for FRAME3  
(Sheet 6 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
FA	I	2	1	PRLD	2
FA3	R	4	1	PRFBKM	4
FACE	I	2	1	COLOR	2
FACE	I	2	1	TSPD1	2
FACEDP	I	2	128	WORK4	256
FACEL	I	4	256	NSTABL	1024
FACELF	I	2	1	NSEDGE	2
FACEN	I	2	1	TSPD1N	2
FACER	I	4	256	NSTABL	1024
FACERT	I	2	1	NSEDGE	2
FACESP	R	4	2	NSEDGE	8
FACEV	I	2	600	WORK4	1200
FACEXT	I	2	600	WORK4	1200
FACLUS	I	2	128	WORK4	256
FACNCL	I	2	128	WORK4	256
FACNED	I	2	128	WORK4	256
FACOLR	I	2	128	WORK4	256
FACTEX	I	2	128	WORK4	256
FADFLG	L	4	1	OPTNS	4
FANUM	I	2	1	WORK11	2
FB	I	2	270	PRVP	540
FBITS	I	2	1	WORK11	2
FBKENB	I	2	1	PRFBKD	2
FBKENT	I	2	1	PRFBKD	2
FBKWBT	I	2	1	PRFBKD	2
FBKWTP	I	2	1	PRFBKD	2
FBLN	I	2	1	WORK11	2
FBLEND	I	2	128	WORK4	256
FBSUN	I	2	1	COLOR	2
FBTM	I	4	1	PRFMOD	4
FCENX	R	4	1	WORK11	4
FCENY	R	4	1	WORK11	4
FCENZ	R	4	1	WORK11	4
FCLUST	I	2	1	WORK11	2
FCOEF	R	4	3	VPFADE	12
FCOLR	I	2	1	WORK11	2
FDSLCT	I	2	3	PROUT1	6
FE	I	2	1	TSEDGF	2
FEAT	I	2	1	COLOR	2
FEATN	I	2	1	WORK11	2
FEDGPT	I	2	1	WORK11	2
FEFLG	I	2	1	PREDGA	2
FELE	I	2	1	TSBNSF	2
FEM	I	2	18	PRESM	36
FEME	I	2	1	TSBNSF	2
FFA	R	4	1	VPFADE	4
FFB	R	4	1	VPFADE	4
FHEIGH	R	4	1	WORK11	4
FIB	R	4	256	PRFBKM	1024
FIBFO	R	4	1	PRFONM	4
FIL	R	4	1	ARECAL	4
FIL	R	4	4	PRLD	16
FIL	R	4	1	PTLSIT	4

Table C-4: Variable List  
for FRAME3  
(Sheet 7 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
FILE	R	4	1	PREDGE	4
FILFIR	R	4	1	ARIN	4
FILM	R	4	18	PRESM	72
FILP	R	4	1	ARECAL	4
FILSZ	R	4	1	ARECAL	4
FIR	R	4	4	PRLD	16
FIRE	R	4	1	PREDGE	4
FIRM	R	4	18	PRESM	72
FIT	R	4	256	PRFBKM	1024
FITFO	R	4	1	PRFONM	4
FJL	R	4	1	ARECAL	4
FJL	R	4	1	PRAREA	4
FJLP	R	4	1	ARECAL	4
FJLPZ	R	4	2	LOCAL	8
FJN	R	4	1	PRAUPD	4
FJR	R	4	1	PRAREA	4
FL	I	4	4	PRLD	16
FLE	I	2	1	TSEDGF	2
FLEX	I	2	1	TSBNSF	2
FLM	I	4	18	PRESM	72
FLONGX	R	4	1	WORK11	4
FLONGY	R	4	1	WORK11	4
FLONGZ	R	4	1	WORK11	4
FMC	R	4	3	VPCFC	12
FNEDGE	I	2	1	WORK11	2
FNORMX	R	4	1	WORK11	4
FNORMY	R	4	1	WORK11	4
FNORMZ	R	4	1	WORK11	4
FNXTCL	I	2	1	WORK11	2
FOPG	R	4	1	VPFM	4
FOPS	R	4	1	VPFM	4
FORI	R	4	1	EDGGEN	4
FORI	R	4	1	EDGORD	4
FORI	R	4	1	INIT3	4
FORI	R	4	1	NSRSLV	4
FORI	R	4	1	PARSEL	4
FORI	R	4	1	PATPRO	4
FORI	R	4	1	PRINIT	4
FORI	R	4	1	TSEA	4
FORI	R	4	1	WNDDMP	4
FORI1	R	4	1	TSBSNO	4
FORI1	R	4	1	TSDBN	4
FORI1	R	4	1	TSEMOV	4
FORI1	R	4	1	TSESP	4
FORI1	R	4	1	TSLODS	4
FORI1X	R	4	1	TSEMOV	4
FORI1X	R	4	1	TSESP	4
FORI2	R	4	1	TSBSNO	4
FORI2	R	4	1	TSDBN	4
FORI2	R	4	1	TSLODS	4
FORI2X	R	4	1	TSEMOV	4
FORI2X	R	4	1	TSESP	4
FORIE	R	4	1	PATPRO	4

Table C-4: Variable List  
for FRAME3  
(Sheet 8 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
FORIG	R	4	1	PATPRO	4
FORII	R	4	1	PRELOD	4
FORIP	R	4	1	TSESP	4
FORIS	R	4	1	TSESP	4
FORIT	R	4	1	NSRSLV	4
FORJ	R	4	1	INIT3	4
FORJ	R	4	1	NSRSLV	4
FORJ	R	4	1	TSSHAD	4
FORJ	R	4	1	VPSIMP	4
FORJJ	R	4	1	TSINIT	4
FORK	R	4	1	EDGORD	4
FORK	R	4	1	NSRSLV	4
FORK	R	4	1	TSEDA	4
FORK	R	4	1	VIDPRO	4
FORK	R	4	1	VPSIMP	4
FORK	R	4	1	VPTEX	4
FORK	R	4	1	WNDDMP	4
FORKK	R	4	1	TSINIT	4
FORLL	R	4	1	TSINIT	4
FORN	R	4	1	WNDDMP	4
FOUND	L	4	1	COLOR	4
FP	I	2	12	DECODE	24
FPRIO	I	2	1	WORK11	2
FR	I	4	4	PRLD	16
FRAD	R	4	1	WORK11	4
FRB	I	4	3	PRLD	12
FRBM	I	4	18	PRESM	72
FRFOB	I	2	1	PRFONM	2
FRFOT	I	2	1	PRFONM	2
FRM	I	4	18	PRESM	72
FRT	I	4	3	PRLD	12
FRTM	I	4	18	PRESM	72
FSHORT	R	4	1	WORK11	4
FSURF	R	4	1	VPFADE	4
FSZ	R	4	1	ARECAL	4
FSZ	R	4	1	PTLSIT	4
FTOP	I	4	1	PRFMOD	4
FVISED	I	4	1	PRCTRL	4
FVORD	I	2	128	WORK4	256
FVPG	R	4	1	VPFM	4
FVPS	R	4	1	VPFM	4
FWPG	R	4	1	VPFM	4
FWPS	R	4	1	VPFM	4
GCOL	I	4	6	RAMCOM	24
GND	I	2	3	VPFM	6
HALFNA	I	2	92	NSOUT	184
HALFNA	I	2	94	PROUT	188
HAZCOR	R	4	1	VPCFC	4
HAZG	I	2	3	VPFM	6
HAZS	I	2	3	VPFM	6
HBB	I	2	1	PRFONM	2
HBT	I	2	1	PRFONM	2
HDRSW	I	2	1	NSEDGE	2

Table C-4: Variable List  
for FRAME3  
(Sheet 9 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
HEF	I	2	3	TSPD1	6
HEFN	I	2	1	TSPD1N	2
HFIB	R	4	1	PRFONM	4
HFIT	R	4	1	PRFONM	4
HFLE	I	2	1	TSEDGF	2
HFLEX	I	2	1	TSBNSF	2
HFOV	R	4	1	FRM1	4
HFRFOB	I	2	1	PRFONM	2
HFRFOT	I	2	1	PRFONM	2
HHDR	I	2	1800	WNDDMP	3600
HICLUS	C	1	1	WORK4	1
HMLFOB	I	2	1	PRFONM	2
HMLFOT	I	2	1	PRFONM	2
HMPFOB	I	2	1	PRFONM	2
HMPFOT	I	2	1	PRFONM	2
HMRFOB	I	2	1	PRFONM	2
HMRFOT	I	2	1	PRFONM	2
HNA	I	2	92	VPAINC	184
HOSFOB	I	2	1	PRFONM	2
HOSFOT	I	2	1	PRFONM	2
HRZFLG	I	4	1	PROUT	4
HTSFOB	I	2	1	PRFONM	2
HTSFOT	I	2	1	PRFONM	2
HUB	I	2	1	PRFONM	2
HUT	I	2	1	PRFONM	2
I	I	2	1	COLOR	2
I	I	4	1	CXMAP	4
I	I	4	1	DECODE	4
I	I	4	1	EDGGEN	4
I	I	4	1	EDGORD	4
I	I	4	1	INIT3	4
I	I	4	1	MODSET	4
I	I	4	1	MODULA	4
I	I	4	1	NSOUT	4
I	I	4	1	NSRSLV	4
I	I	4	1	ORDER	4
I	I	4	1	PARSEL	4
I	I	4	1	PATPRO	4
I	I	4	1	PRCLR	4
I	I	4	1	PRDMP	4
I	I	4	1	PRINIT	4
I	I	4	1	PRIRSV	4
I	I	4	1	PROUT	4
I	I	4	1	PTLGEN	4
I	I	4	1	PTLSIT	4
I	I	4	1	PUT	4
I	I	4	1	RAMOUT	4
I	I	4	1	RAMSET	4
I	I	4	1	REED	4
I	I	4	1	STPED	4
I	I	4	1	STPLT	4
I	I	4	1	TSEA	4
I	I	4	1	TSEDGR	4

Table C-4: Variable List  
for FRAME3  
(Sheet 10 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
I	I	4	1	VPLLI	4
I	I	4	1	VPPTR	4
I	I	4	1	VPSIMP	4
I	I	4	1	WNDDMP	4
IO	I	4	1	FRM1	4
IO	I	4	1	VPLLI	4
I1	I	4	1	ARIN	4
I1	I	4	1	NSRSLV	4
I1	I	4	1	TSBSNO	4
I1	I	4	1	TSDBN	4
I1	I	4	1	TSEMOV	4
I1	I	4	1	TSESP	4
I1	I	4	1	TSLODS	4
I1	I	4	1	TSPINC	4
I12	I	4	1	PROUT	4
I1X	I	4	1	TSEMOV	4
I1X	I	4	1	TSESP	4
I2	I	4	1	NSRSLV	4
I2	I	4	1	TSBSNO	4
I2	I	4	1	TSDBN	4
I2	I	4	1	TSEMOV	4
I2	I	4	1	TSESP	4
I2	I	4	1	TSLODS	4
I2	I	4	1	TSPINC	4
I2X	I	4	1	TSEMOV	4
I2X	I	4	1	TSESP	4
I3	I	4	1	TSPINC	4
IABSAD	I	4	1	MODSET	4
IABSAD	I	4	1	SETRD	4
IAC	I	4	1	PRESEL	4
IALL	I	4	1	PRTPLU	4
IALR	I	4	1	PRTPLU	4
IAOB	I	2	90	PRVP	180
IAOBI	I	4	1	VPMLF	4
IAP	I	4	1	PRAPLU	4
IAP1	I	4	1	PREDGA	4
IAP2	I	4	1	PREDGA	4
IAPB	I	4	1	PREDGA	4
IAPB1	I	4	1	PREDGA	4
IAPB2	I	4	1	PREDGA	4
IAPC	I	4	1	PREDGA	4
IAPT	I	4	1	PREDGA	4
IAPT1	I	4	1	PREDGA	4
IAPT2	I	4	1	PREDGA	4
IARG	I	4	1	MODSET	4
IARG	I	4	1	PUT	4
IARG	I	4	1	REED	4
IARG	I	4	1	SETFIL	4
IARG	I	4	1	SETRD	4
IARL	I	4	1	PRTPLU	4
IARR	I	4	1	PRTPLU	4
IB	I	4	30	CXMAP	120
IB	R	4	1	EDGGEN	4

Table C-4: Variable List  
for FRAME3  
(Sheet 11 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
IB	I	2	4	PRLD	8
IB	I	4	30	STPED	120
IB	I	4	1	STPLT	4
IBA	I	2	1	PREDGA	2
IBC	I	4	1	PRESEL	4
IBF	I	4	1	ARECAL	4
IBF	I	4	1	EDGGEN	4
IBKG	I	4	1	VPILN	4
IBL	I	4	1	TSPINC	4
IBLU	I	2	3	PROUT1	6
IBM	I	2	18	PRESM	36
IBMF	I	4	1	NSEDGR	4
IBOTH	I	4	1	DECODE	4
IBR	I	4	1	TSPINC	4
IBTM	I	2	1	PREDGA	2
IBTMTP	I	2	8	PRTPL	16
IBX	I	4	1	TSBNST	4
IC	I	2	1	ARIN	2
IC	I	4	1	TSBSNO	4
IC	I	4	1	TSDBN	4
ICAS	I	4	1	TSPINC	4
ICASE	I	2	1	ARIN	2
ICHAN	I	4	1	OPTNS	4
ICLOS	I	4	1	CXMAP	4
ICLOS	I	4	1	EDGGEN	4
ICLOS	I	4	1	INIT3	4
ICLOS	I	4	1	PRIRSV	4
ICLOS	I	4	1	PTLGEN	4
ICLOS	I	4	1	PTLSIT	4
ICLOS	I	4	1	STPED	4
ICLOS	I	4	1	STPLT	4
ICLOS	I	4	1	VIDPRO	4
ICLRIX	I	4	1	PRTPLU	4
ICNT	I	4	1	FRAME3	4
ICNT	I	4	1	NSRSLV	4
ICNTR	I	4	1	EDGGEN	4
ICNTR	I	4	1	PTLGEN	4
ICOSYS	I	4	1	MISC	4
ICURR	I	2	1	TSCTRL	2
IDADJO	I	2	1	ARIN	2
IDEF	I	2	1	ARIN	2
IDEV	I	4	1	DEV	4
IE	I	4	1	PATPRO	4
IED	I	4	1	FRM3	4
IEF	I	4	1	MISC	4
IEFLG	I	4	100	STRIP	400
IEHDR	I	4	400	STRIP	1600
IELAP	I	4	1	FRAME3	4
IELSE	I	2	1	DECODE	2
IEPRI	I	4	1	PREDGA	4
IEQ	I	4	1	PRTPLU	4
IESMA	I	4	8	PRESMI	32
IESMB	I	4	8	PRESMI	32

Table C-4: Variable List  
for FRAME3  
(Sheet 12 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
IFACE	I	4	1	ARIN	4
IFACE2	I	4	1	ARIN	4
IFACEL	I	4	1	PREDGE	4
IFACER	I	4	1	PREDGE	4
IFACL	I	4	200	GEN	800
IFACR	I	4	200	GEN	800
IFACX	I	4	1	EDGORD	4
IFBKB	I	4	1	PRFBKD	4
IFBKT	I	4	1	PRFBKD	4
IFD	I	4	1	PREDGA	4
IFLG	I	2	1	EDGGEN	2
IFLG	I	2	1	FRAME3	2
IFLG	I	2	1	INIT3	2
IFLG	I	4	1	PRIRSV	4
IFLG	I	2	1	PTLSIT	2
IFLG	I	4	1	STPED	4
IFLG	I	2	1	STPLT	2
IFLG	I	2	1	WNDDMP	2
IFLS	I	4	1	NSRSLV	4
IFOB	I	4	1	PRFONM	4
IFOT	I	4	1	PRFONM	4
IFPRI	I	2	4096	FACPR	8192
IFRBTM	I	4	256	PRFBKM	1024
IFRS	I	4	1	NSRSLV	4
IFRTOP	I	4	256	PRFBKM	1024
IFS	I	4	1	VPFADE	4
IFXLOD	I	4	1	OPTNS	4
IGRN	I	2	3	PROUT1	6
IHDT	I	4	1	FRM3	4
IHORIZ	I	2	1	VPFDC	2
IHP1	I	4	1	PREDGA	4
IHP2	I	4	1	PREDGA	4
IHP3	I	4	1	PREDGA	4
IHRZ	I	4	1	NSEDGR	4
II	I	4	1	ARIN	4
II	I	4	1	PRELOD	4
II	I	4	1	TSEDA	4
II	I	4	1	TSEMOV	4
II	I	4	1	TSINIT	4
II	I	4	1	TSTXMD	4
IJ	I	4	1	TSTXMD	4
IJ	I	4	1	WNDDMP	4
IJLE	I	4	1	PREDGA	4
IJLM	I	4	18	PRESM	72
IJRE	I	4	1	PREDGA	4
IK	I	4	1	TSESP	4
IL	I	4	1	CXMAP	4
IL	R	4	100	STRIP	400
IL	I	2	1	TSCTRL	2
ILE	I	4	1	PRFBKU	4
ILEFT	I	4	1	TSTXMD	4
ILFAC	I	4	1	EDGGEN	4
ILNE	I	2	2	TSBNSF	4

Table C-4: Variable List  
for FRAME3  
(Sheet 13 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
ILOC	I	4	1	NSRSLV	4
ILOD	I	4	1	TSTXMD	4
ILOD1	I	4	1	TSTXMD	4
ILT	I	4	1	FRM3	4
IMD	I	4	1	PREDGA	4
IMIN	I	4	1	FRAME3	4
IMOD	I	2	1	ARIN	2
IN	I	4	1	AREA2	4
IN	I	4	1	SINGS	4
IN4	I	4	1	EDGGEN	4
INB	I	4	1	ARIN	4
INC	I	4	1	STPED	4
INC	I	4	1	STPLT	4
INC	I	2	2	TSBNSF	4
INDEX	I	4	1	LR2	4
INDEX	I	4	1	SINGS	4
INDEX	I	4	1	TB2	4
INDXE	I	4	1	VPPTR	4
INDXL	I	4	1	VPPTR	4
INEW	I	4	1	PRESMI	4
INIBF2	I	2	4	RAMSET	8
INIBF4	I	4	2	RAMSET	8
INITFR	I	4	1	PRCTRL	4
INK1	I	4	1	TSTXMD	4
INK2	I	4	1	TSTXMD	4
INT	I	4	1	ARECAL	4
INT	I	4	1	PRESEL	4
INT	I	4	1	PRIRSV	4
INT	I	4	1	PRTPLU	4
INT	I	4	1	PTLSIT	4
INT	I	4	1	TSEA	4
IO	I	4	1	ARIN	4
IOBJ	I	4	1	ARIN	4
IP	I	4	1	DECODE	4
IP	I	4	1	TSESP	4
IP	I	4	1	TSTXMD	4
IP1	I	4	1	TSEDA	4
IP1A	I	4	1	PRESMI	4
IP1B	I	4	1	PRESMI	4
IP2	I	4	1	TSEDA	4
IP2A	I	4	1	PRESMI	4
IP2B	I	4	1	PRESMI	4
IP3	I	4	1	TSEDA	4
IP4	I	4	1	TSEDA	4
IPATH	I	4	1	TSMUX	4
IPES	I	4	2	TSEDGF	8
IPFLG	I	4	1	PRCTRL	4
IPLGER	I	4	1	PREPD	4
IPLQL	I	4	1	PRTPLU	4
IPLQR	I	4	1	PRTPLU	4
IPP1	I	4	1	TSTXMD	4
IPR	I	4	4	PRESMI	16
IPRB1	I	4	1	PREDGA	4

Table C-4: Variable List  
for FRAME3  
(Sheet 14 of 32)



SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
ISTAT	I	4	1	SETFIL	4
ISTAT	I	4	1	SETRD	4
ISTAT	I	4	1	VPLTC	4
ISTIM	I	4	3	FRAME3	12
ISWP	I	4	1	ORDER	4
IT	R	4	1	EDGGEN	4
IT	I	4	1	NSRSLV	4
IT	I	2	4	PRLD	8
ITA	I	2	1	PREDGA	2
ITB	I	4	1	ARECAL	4
ITC	I	4	1	TSTXMD	4
ITEMP	I	4	46	WNDDMP	184
ITEXC	I	4	100	STRIP	400
ITF	I	4	1	ARECAL	4
ITF	I	4	1	EDGGEN	4
ITIM	I	4	3	FRAME3	12
ITL	I	4	1	TSPINC	4
ITM	I	4	8	FRM3	32
ITM	I	2	18	PRESM	36
ITMP	I	4	1	EDGORD	4
ITMP	I	4	1	PRIRSV	4
ITOP	I	2	1	PREDGA	2
ITOPTP	I	2	8	PRTPL	16
ITORB	I	2	1	PRTPLU	2
ITP1	I	4	1	PREDGA	4
ITP2	I	4	1	PREDGA	4
ITP3	I	4	1	PRTPLU	4
ITPF	I	4	1	NSEDGR	4
ITPK	I	4	1	PRTPLU	4
ITPRIL	I	4	8	PRTPL	32
ITPRIR	I	4	8	PRTPL	32
ITR	I	4	1	TSPINC	4
ITXS	I	4	1	TSTXMD	4
IU	I	4	1	TSBSNO	4
IU	I	4	1	TSDBN	4
IU	I	4	1	TSEMOV	4
IU	I	4	1	TSLODS	4
IU	I	4	1	TSTXMD	4
IU1	I	4	1	TSBSNO	4
IU1	I	4	1	TSEDA	4
IU1	I	4	1	TSLODS	4
IUP	I	4	1	MODSET	4
IUP	I	4	1	PUT	4
IUP	I	4	1	REED	4
IV1	I	2	1	DECODE	2
IV2	I	2	1	DECODE	2
IV3	I	2	1	DECODE	2
IWL	I	4	1	NSEDGE	4
IX	I	4	5	BFRI	20
IX	I	4	5	BFRM	20
IX	I	4	5	BFRO	20
IXCOL	I	4	1	FRM3	4
IXP	I	4	1	TSTXMD	4

Table C-4: Variable List  
for FRAME3  
(Sheet 16 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
IXTM	I	4	6	TXMAPS	24
IXX	I	4	1	STPLT	4
J	I	4	1	CXMAP	4
J	I	4	1	DECODE	4
J	I	4	1	EDGGEN	4
J	I	4	1	INIT3	4
J	I	4	1	NSOUT	4
J	I	4	1	NSRSLV	4
J	I	4	1	ORDER	4
J	I	4	1	PRDMP	4
J	I	4	1	PROUT	4
J	I	4	1	PTLGEN	4
J	I	4	1	PTLSIT	4
J	I	4	1	STPED	4
J	I	4	1	STPLT	4
J	I	4	1	TSEDGR	4
J	I	4	1	TSMUX	4
J	I	4	1	TSSHAD	4
J	I	4	1	VPLLI	4
J	I	4	1	VPPTR	4
J	I	4	1	VPSIMP	4
JO	I	4	1	FRM1	4
JO	I	4	1	VPLLI	4
J1	I	4	1	MODSET	4
J1	I	4	1	PRDMP	4
J1	I	4	1	PUT	4
J1	I	4	1	REED	4
J2	I	4	1	PRDMP	4
J3	I	4	1	PRDMP	4
JAE	I	4	1	VPPTR	4
JAL	I	4	1	VPPTR	4
JARG	I	4	1	MODSET	4
JARG	I	4	1	PUT	4
JARG	I	4	1	REED	4
JARG	I	4	1	SETRD	4
JC1	R	4	1	EDGGEN	4
JC2	R	4	1	EDGGEN	4
JCURR	I	2	1	TSCTRL	2
JDADJO	I	2	1	ARIN	2
JE	I	4	90	PRVP	360
JED	I	4	1	PROUT2	4
JEFLG	I	4	200	GEN	800
JEHDR	I	4	800	GEN	3200
JEL	I	4	1	JWIN	4
JEL	I	4	1	TSBSNO	4
JER	I	4	1	JWIN	4
JHDR	I	4	900	WNDDMP	3600
JINDX	I	2	512	LTORD	1024
JINIT	I	4	1	VPTEX	4
JJ	I	4	1	TSINIT	4
JL	R	4	100	STRIP	400
JL	I	2	3	TSPD1	6
JLEFT	I	4	256	NSTABL	1024

Table C-4: Variable List  
for FRAME3  
(Sheet 17 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
-----	-	-	-----	-----	-----
JLIM	I	4	1	VIDPRO	4
JLIT	I	4	40	PLGVP	160
JLN	I	2	1	TSPD1N	2
JLP	R	4	1	EDGGEN	4
JLS	I	4	1	NSRSLV	4
JLS	I	4	1	TSEMOV	4
JLS	I	4	300	WNDDMP	1200
JMAX	I	4	1	TSEA	4
JMAX	I	4	1	TSEDGR	4
JMOD	I	4	1	PATPRO	4
JMP1	I	4	1	TSEA	4
JN	I	4	1	PRLD	4
JN	I	2	2	TSINC	4
JND	I	4	1	TSDBN	4
JNEXT	I	2	1	TSCTRL	2
JNM1	I	2	2	TSINC	4
JPR	I	2	2	TSINC	4
JPREV	I	2	1	TSCTRL	2
JPROC	I	4	1	MODSET	4
JR	R	4	100	STRIP	400
JR	I	2	3	TSPD1	6
JRCYC	I	4	1	TSCTRL	4
JRCYCS	I	4	1	VPTEX	4
JREC	I	4	1	MODSET	4
JREC	I	4	1	REED	4
JRFC	I	4	1	MODSET	4
JRN	I	2	1	TSPD1N	2
JRP	R	4	1	EDGGEN	4
JSAP	I	4	1	VPTEX	4
JSSW	I	4	1	MISC	4
JSTR	I	4	1	TSCTRL	4
JTEXC	I	4	200	GEN	800
JX	I	4	1	MODSET	4
JX	I	4	1	PUT	4
JX	I	4	1	REED	4
K	I	4	1	CXMAP	4
K	I	4	1	DECODE	4
K	I	4	1	EDGGEN	4
K	I	4	1	EDGORD	4
K	I	4	1	MODSET	4
K	I	4	1	NSRSLV	4
K	I	4	1	ORDER	4
K	I	4	1	PRAPLU	4
K	I	4	1	PRAUPD	4
K	I	4	1	PRDMP	4
K	I	4	1	PRELOD	4
K	I	4	1	PRTPLU	4
K	I	4	1	PTLGEN	4
K	I	4	1	PTLSIT	4
K	I	4	1	PUT	4
K	I	4	1	REED	4
K	I	4	1	TSDBN	4
K	I	4	1	TSEDA	4

Table C-4: Variable List  
for FRAME3  
(Sheet 18 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
K	I	4	1	TSEDGR	4
K	I	4	1	TSESP	4
K	I	4	1	TSMUX	4
K	I	4	1	TSSHAD	4
K	I	4	1	TSTXMD	4
K	I	4	1	VIDPRO	4
K	I	4	1	VPAINC	4
K	I	4	1	VPCFC	4
K	I	4	1	VPFADE	4
K	I	4	1	VPLTC	4
K	I	4	1	VPMLF	4
K	I	4	1	VPSIMP	4
K	I	4	1	WNDDMP	4
K1	I	4	1	SAVELT	4
K1	I	4	1	VIDPRO	4
K1	I	4	1	VPLTC	4
K2	I	4	1	SAVELT	4
K2	I	4	1	VPLTC	4
K3	I	4	1	SAVELT	4
K3	I	4	1	VPLTC	4
KARG	I	4	1	MODSET	4
KARG	I	4	1	PUT	4
KB	I	4	1	DEVCOM	4
KE	I	4	1	EDGORD	4
KEDG	I	4	1	GEN	4
KEDGM	I	4	1	EDGGEN	4
KEG	I	4	1	WNDDMP	4
KGND	R	4	1	VPFM	4
KI	R	4	1	FRM1	4
KIJ	R	4	4	FIXDT	16
KINDX	I	2	512	EDORD	1024
KJ	R	4	1	FRM1	4
KJ	I	4	1	VPMLF	4
KJ	I	4	1	VPSIMP	4
KK	I	4	1	TSINIT	4
KLFAC	I	4	400	GEN	1600
KLHDR	I	4	1600	GEN	6400
KLIM	I	4	1	EDGGEN	4
KLIM	I	4	1	EDGORD	4
KLIT	I	4	1	GEN	4
KLOD	I	4	1	FIXDT	4
KLTAB	R	4	16	FIXDT	64
KNT	I	4	1	STPED	4
KNT	I	4	1	STPLT	4
KP	I	4	1	PRAUPD	4
KPX	R	4	3	WORK11	12
KPY	R	4	3	WORK11	12
KPZ	R	4	3	WORK11	12
KRASH	I	2	1	VPFM	2
KRFAC	I	4	1	EDREL	4
KS	R	4	1	FRM1	4
KSKY	R	4	1	VPFM	4
KUVW	R	4	6	FIXDT	24

Table C-4: Variable List  
for FRAME3  
(Sheet 19 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
L	I	4	1	CXMAP	4
L	I	2	1	DECODE	2
L	I	4	1	NSRSLV	4
L	I	4	1	ORDER	4
L	I	4	1	PRDMP	4
L	I	4	1	PTLGEN	4
L	I	4	1	TSEDA	4
L	I	4	1	TSESP	4
L	I	4	1	TSTXMD	4
LO	I	4	1	VPLLI	4
LO110	I	2	1	PREEFS	2
L1	I	4	1	TSESP	4
L1001	I	2	1	PREEFS	2
L2	I	4	1	TSESP	4
L3	R	4	512	ORDER	2048
L3	I	4	1	TSESP	4
LADR	I	4	1	VPLTC	4
LARG	I	4	1	MOLSET	4
LAYREC	I	2	1	WORK4	2
LCOL	I	4	400	GEN	1600
LDA	I	2	1	PREDGA	2
LDB	I	2	1	PREDGA	2
LE	I	2	1	TSEDGF	2
LE	I	2	1584	WORK	3168
EC	I	4	1	VPMLF	4
LEUMP	I	2	1	FRM3	2
LEDG	I	4	1	NSOUT	4
LEDG	I	4	1	PATPRO	4
LEDG	I	4	1	PROUT	4
LEFT	I	4	1	MODSET	4
LEFT	I	4	1	PUT	4
LEFT	I	4	1	REED	4
LEFT	I	4	1	TB2	4
LEFT	I	4	1	WNDDMP	4
LEND	I	2	1	PRIRSV	2
LENE	I	2	1	TSBNSF	2
LEOFS	I	4	1	VIDPRO	4
LEOFS	I	4	1	VPMLF	4
LEOFS	I	4	1	VPSIMP	4
LET	I	2	1	VPMLF	2
LET	I	2	1	VPSIMP	2
LETOT	I	4	1	VPPTR	4
LFAC	I	4	1	NSRSLV	4
LFACN	I	4	50	PTLSET	200
LFDT	I	4	2	FRM3	8
LFREEZ	I	2	1	WORK4	2
LHDR	I	4	200	PTLSET	800
LI	I	4	1	VPLLI	4
LIM	I	4	1	WNDDMP	4
LIMED	I	4	1	VPAINC	4
LIMEDG	I	4	1	STPED	4
LIMLIT	I	4	1	STPLT	4
LIMLT	I	4	1	VPLTC	4

Table C-4: Variable List  
for FRAME3  
(Sheet 20 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
LITC	I	4	50	PTLSET	200
LITLIM	I	4	1	PTLGEN	4
LITLUN	I	4	2	PTLNAM	8
LITSZ	R	4	50	PTLSET	200
LJ	I	4	1	VPLLI	4
LL	I	4	1	TSINIT	4
LLDT	I	4	2	FRM3	8
LLFENB	I	2	1	PRFBKD	2
LLFENT	I	2	1	PRFBKD	2
LLFLG	I	4	1	VPLLI	4
LLIM	I	4	1	SAVELT	4
LLOC	I	4	1	NSRSLV	4
LMEM1	I	4	1	LTC	4
LMEM1	I	4	1	VPAINC	4
LMIN	I	4	1	VPLLI	4
LMLM1	I	4	1	VPLTC	4
LN	I	4	1	MISC	4
LNA	I	4	1	PTLSIT	4
LNB	I	4	1	FRM3	4
LNC	I	4	1	EDGGEN	4
LNE	I	4	1	FRM3	4
LNSP	R	4	1	EDGGEN	4
LNST	R	4	1	EDGGEN	4
LO	I	4	1	MISC	4
LOB	I	2	2	DCOUT	4
LGC	I	4	1	NSRSLV	4
LOCFLG	L	4	1	OPTNS	4
LOD	I	4	12	TSLODV	48
LOD	C	1	1	WORK4	1
LODF	I	4	13	TSTXMD	52
LODMOD	L	4	1	OPTNS	4
LODS	I	4	1	TSPINC	4
LODT	I	4	16	TSLOD	64
LOWPRI	I	4	1	PRTPLU	4
LP	I	4	1	DEVCOM	4
LPA	I	2	90	PRVP	180
LPB	I	2	90	PRVP	180
LPCT	I	4	1	MODSET	4
LPCT	I	4	1	PUT	4
LPCT	I	4	1	REED	4
LPL	I	4	40	PLGVP	160
LPLT	I	4	1	PTLGEN	4
LPN	I	4	1	PTLGEN	4
LPRA	I	4	1	PRTPLU	4
LRF	I	2	2	DCOUT	4
LSB	I	4	1	TSTXMD	4
LSBM	I	4	1	TSTXMD	4
LSBSM	I	4	2	TSTXMD	8
LSBX	I	4	3	TSTXMD	12
LSIZ	I	4	40	PLGVP	160
LSP	I	4	1	MISC	4
LST	I	4	1	MISC	4
LSTLEN	I	2	1	PREDGA	2

Table C-4: Variable List  
for FRAME3  
(Sheet 21 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
LSZP	I	4	1	PTLSIT	4
LTCMP	I	2	1	FRM3	2
LTLIM	I	4	1	PTLSIT	4
LTOT	I	4	1	VPPTR	4
LTPR	I	2	2	PROUT2	4
LUNE	I	4	1	VPPTR	4
LUNED	I	2	2	LUNEDG	4
LUNL	I	4	1	VPPTR	4
LUTSE2	I	2	4	RAMSET	8
LUTSE4	I	4	2	RAMSET	8
LZE	I	4	1	FRM3	4
LZL	I	4	1	FRM3	4
M	I	4	1	CXMAP	4
M	I	4	1	ORDER	4
M	I	4	1	PRDMP	4
M	I	4	1	PUT	4
M	I	4	1	STPED	4
M	I	4	1	STPLT	4
M	I	4	1	TSESP	4
M	I	4	1	WNDDMP	4
M1	I	4	1	TSTXMD	4
M2	I	4	1	TSTXMD	4
M3	I	4	1	TSTXMD	4
MAP	I	2	2048	CXMAP	4096
MAPA	I	4	1024	CXMAP	4096
MAPA	I	4	1024	STPED	4096
MAPA	I	4	1024	STPLT	4096
MAPC	I	2	6	TSPD2	12
MAPE	I	2	2048	STPED	4096
MAPL	I	2	2048	STPLT	4096
MAPSET	I	4	1	TSTXMD	4
MAX	I	4	1	VIDPRO	4
MAXDIS	R	4	64	WORK17	256
MAXLOD	I	4	1	TSLODS	4
MAXLYR	I	4	1	WORK21	4
MAXPRI	I	4	1	PRAPLU	4
MAXRNG	R	4	8	MISC	32
MAXTP	I	4	1	PRTPLU	4
MB	I	2	270	PRVP	540
MBK	I	4	1	VPCFC	4
MC	I	4	6	TXMAPS	24
MCP	I	4	1	TSTXMD	4
MDFNO	I	4	1	LR2	4
MDFNO	I	4	1	SINGS	4
MDFNO	I	4	1	TB2	4
NDSLCT	I	2	3	PROUT1	6
ME	I	2	1	TSEDGF	2
MELE	I	2	1	TSBNSF	2
MEME	I	2	1	TSBNSF	2
NIN	I	4	1	VIDPRO	4
MINRNG	R	4	8	MISC	32
NJ	I	4	1	VPMLF	4
MJ	I	4	1	VPSIMP	4

Table C-4: Variable List  
for FRAME3  
(Sheet 22 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
-----	-----	-----	-----	-----	-----
MK	R	4	1	FIXDT	4
ML	I	2	3	PRLD	6
MLFBB	I	2	256	PRFBKM	512
MLFBT	I	2	256	PRFBKM	512
MLFLG	I	4	1	PRTPLU	4
MLFOB	I	2	1	PRFONM	2
MLFOT	I	2	1	PRFONM	2
MLM	I	2	18	PRESM	36
MM	I	2	12288	TXMAPS	24576
MMPOS	L	4	1	OPTNS	4
MMW	I	4	6144	TXMAPS	24576
MNEG	I	4	1	VPSIMP	4
MODATA	I	4	1	TSESP	4
MODFLG	I	2	1	PROUT1	2
MODJ	I	4	1	TSPINC	4
MODL	I	2	1	PREDGE	2
MODL	I	2	3	TSPD1	6
MODLFT	I	2	2	PROUT1	4
MODLN	I	2	1	TSPD1N	2
MODR	I	2	1	PREDGE	2
MODR	I	2	3	TSPD1	6
MODRN	I	2	1	TSPD1N	2
MP	I	2	3	PRLD	6
MP	I	4	6	TSTXMD	24
MPBIT	L	4	1	EDGORD	4
MPFBB	I	2	256	PRFBKM	512
MPFBT	I	2	256	PRFBKM	512
MPFLG	I	2	1	PREDGE	2
MPFOB	I	2	1	PRFONM	2
MPFOT	I	2	1	PRFONM	2
MPL	I	4	1	TSTXMD	4
NPLOD	I	4	14	TSTXMD	56
MPM	I	2	18	PRESM	36
MPR	I	4	1	TSTXMD	4
MR	I	2	3	PRLD	6
MRFBB	I	2	256	PRFBKM	512
MRFBT	I	2	256	PRFBKM	512
MRFLG	I	4	1	PRTPLU	4
MRFOB	I	2	1	PRFONM	2
MRFOT	I	2	1	PRFONM	2
MRM	I	2	18	PRESM	36
MSKE	I	4	1	CXMAP	4
MSKE	I	4	1	STPED	4
MSKL	I	4	1	CXMAP	4
MSKL	I	4	1	STPLT	4
MXC	I	4	1	TSTXMD	4
MXCT	I	4	1	TSTXMD	4
MXEDG	I	4	1	CTRL	4
MXLIT	I	4	1	CTRL	4
MXLNE	I	4	1	CTRL	4
MXLNT	I	4	1	CTRL	4
MXLODF	I	2	2	TSLODV	4
N	I	4	1	CXMAP	4

Table C-4: Variable List  
for FRAME3  
(Sheet 23 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
N	I	4	1	INIT3	4
N	I	4	1	MODSET	4
N	I	4	1	PRDMP	4
N	I	4	1	PTLGEN	4
N	I	4	1	PUT	4
N	I	4	1	STPED	4
N	I	4	1	STPLT	4
N	I	4	1	VIDPRO	4
N	I	4	1	VPAINC	4
N	I	4	1	VPLTC	4
N	I	4	1	WNDDMP	4
N1	I	4	1	VIDPRO	4
N2	I	2	1	ORDER	2
N2	I	4	1	VIDPRO	4
NA	I	4	8	LOCAL	32
NA	I	4	25	LUNEDG	100
NA	I	4	46	NSOUT	184
NA	I	4	46	PATPRO	184
NA	I	4	47	PROUT	188
NA	I	4	11	SAVELT	44
NA	I	4	46	VPAINC	184
NA	I	4	12	VPLTC	48
NAF	I	4	1	GEN	4
NAFCL	I	4	100	STRIP	400
NAFCR	I	4	100	STRIP	400
NAME	R	8	2	CXMAP	16
NAME	R	4	2	EDGGEN	8
NAME	R	8	1	PTLSIT	8
NAME	R	8	1	SAVELT	8
NAME	R	8	2	STPED	16
NAME	R	4	2	VPAINC	8
NAME	R	8	1	VPLTC	8
NAME	R	8	1	WNDDMP	8
NAML	R	8	1	STPLT	8
NBYTES	I	4	1	RAMCOM	4
NCOL	I	4	100	STRIP	400
NCULF	C	1	1	WORK4	1
NE	I	4	1	FIXDT	4
NE	I	2	5	PRLD	10
NE	I	2	1	TSEDGF	2
NEC	I	4	1	PATPRO	4
NECNT	I	4	1	PRCTRL	4
NED	I	4	1	FRM3	4
NEDG	I	4	1	STRIP	4
NEDGES	I	2	1	WORK4	2
NEFE	I	2	1	TSBNSF	2
NEFLG	I	4	90	PRVP	360
NEG	I	4	1	PRVP	4
NEHDR	I	4	360	PRVP	1440
NEN	I	4	1	VIDPRO	4
NENE	I	2	1	TSBNSF	2
NEWLST	I	2	1	PROUT1	2
NEWMOD	I	4	1	PROUT1	4

Table C-4: Variable List  
for FRAME3  
(Sheet 24 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
NEWOS	I	2	3	MODOUT	6
NEWTHS	I	2	1	MODOUT	2
NEWTS	I	2	3	MODOUT	6
NFA	I	2	3	MODOUT	6
NFACE	I	4	256	PRFACE	1024
NFACL	I	4	100	STRIP	400
NFACR	I	4	100	STRIP	400
NFACT	I	4	1	EDGORD	4
NFSUM	I	4	1	FIXDT	4
NJ	I	4	1	VPTEX	4
NL	I	4	1	FIXDT	4
NLHDR	I	4	160	PLGVP	640
NLIT	I	4	1	PTLSET	4
NLITE	I	4	1	PLGVP	4
NLT	I	4	1	FRM3	4
NM	I	4	1	INIT3	4
NMA	I	2	3	MODOUT	6
NME	I	2	1	MODOUT	2
NMED	I	4	1	CTRL	4
NMLT	I	4	1	CTRL	4
NMSLCT	I	2	3	MODOUT	6
NOBJ	I	4	1	CXMAP	4
NOCOL	I	2	1	EDGGEN	2
NOCOL	I	2	1	FRAME3	2
NOCOL	I	2	1	INIT3	2
NOCOL	I	2	1	PRIRSV	2
NOCOL	I	2	1	PTLSIT	2
NOCOL	I	2	1	STPLT	2
NOCOL	I	2	1	WNDDMP	2
NOEDG	I	2	1	FRM3	2
NOLIT	I	2	1	FRM3	2
NOSEC	I	4	1	FRAME3	4
NOSSEC	I	4	1	FRAME3	4
NOX	I	2	1	PRNXTO	2
NOXF	I	2	1	CASE2	2
NP1	I	4	1	VPAINC	4
NP1	I	4	1	VPLTC	4
NS	I	2	4	PRLD	8
NSBNX	I	2	1	NSEDGE	2
NSE	I	2	1	PREDGE	2
NSET	I	4	1	VIDPRO	4
NSET	I	4	1	VPCFC	4
NSET	I	4	1	VPFAD	4
NSM	I	2	18	PRESM	36
NTERF	C	1	1	WORK4	1
NTEXC	I	4	90	PRVP	360
NTEXF	C	1	1	WORK4	1
NTNENB	I	2	1	PRNEFS	2
NUMFAC	I	2	1	COLOR	2
NVP	I	4	9	FIXDT	36
NVRTEX	I	2	1	WORK4	2
NXFACE	I	2	1	COLOR	2
NXTBLK	C	1	1	WORK4	1

Table C-4: Variable List  
for FRAME3  
(Sheet 25 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
NXTBTM	I	2	1	PRFMOD	2
NXTE	I	4	1	VPAINC	4
NXTEDG	I	2	1	TSCTRL	2
NXTFAC	I	2	64	WORK4	128
NXTL	I	4	1	VPLTC	4
NXTTOP	I	2	1	PRFMOD	2
O	R	4	1	PRFBKU	4
OFA	I	2	3	MODOUT	6
OLDLST	I	2	1	PROUT1	2
OLDMOD	I	4	1	PROUT1	4
OLDOS	I	2	3	MODOUT	6
QLDP1	I	2	1	PREDGA	2
QLDP2	I	2	1	PREDGA	2
QLDTHS	I	2	1	MODOUT	2
QLDTS	I	2	3	MODOUT	6
QMA	I	2	3	MODOUT	6
QME	I	2	1	MODOUT	2
QMSLCT	I	2	3	MODOUT	6
QRGEDG	I	4	1	PROUT	4
ORIENT	I	2	60	WORK15	120
OS	I	2	3	PRLD	6
OSBT	R	4	1	NSOUT	4
OSFBB	I	2	256	PRFBKM	512
OSFBT	I	2	256	PRFBKM	512
OSFLG	I	2	1	PREDGE	2
OSFOB	I	2	1	PRFONM	2
OSFOT	I	2	1	PRFONM	2
OSM	I	2	18	PRESM	36
P	R	4	1	DECODE	4
P1	R	4	1	TSEDA	4
P1CLPS	I	2	1	PRVIS	2
P1FLG	I	2	1	PREDGA	2
P1UFB	I	2	1	FLGS3	2
P1UP2	I	2	1	FLGS3	2
P2	R	4	1	TSEDA	4
P2FLG	I	2	1	PREDGA	2
P2NE	I	2	1	PRVIS	2
P2UFB	I	2	1	FLGS3	2
P3	R	4	1	TSEDA	4
P3ENB	I	2	1	PRFBKD	2
P3ENT	I	2	1	PRFBKD	2
P3EPAB	I	2	1	PRP3S	2
P3EPAT	I	2	1	PRP3S	2
P3FLG	I	2	1	PREDGA	2
P3P	R	4	1	TSEDA	4
P3SBR	R	4	1	PRTPLU	4
P3SFAC	I	2	8	PRP3S	16
P3SFIB	R	4	8	PRP3S	32
P3SFIT	R	4	8	PRP3S	32
P3SFLG	I	2	1	PRP3S	2
P3SML	I	2	8	PRP3S	16
P3SMP	I	2	8	PRP3S	16
P3SMR	I	2	8	PRP3S	16

Table C-4: Variable List  
for FRAME3  
(Sheet 26 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
P3SOS	I	2	8	PRP3S	16
P3STS	I	2	8	PRP3S	16
P4	R	4	1	TSEDA	4
P4P	R	4	1	TSEDA	4
PA	R	4	6	TSTXEV	24
PARAM	I	4	10	RAMOUT	40
PARAM1	I	4	10	RAMSET	40
PARAM2	I	4	10	RAMOUT	40
PARAM3	I	4	10	RAMOUT	40
PARAM4	I	4	10	RAMSET	40
PARAM5	I	4	10	RAMSET	40
PARAM6	I	4	10	RAMSET	40
PARAM7	I	4	10	RAMSET	40
PBGCNT	I	2	1	PRCTRL	2
PCFIC	I	2	1280	WORK10	2560
PCOLNT	I	2	1280	WORK10	2560
PCOLWT	I	2	1280	WORK10	2560
PE	R	4	1	TSEDA	4
PFACOD	I	2	1280	WORK10	2560
PHEIGH	I	2	1280	WORK10	2560
PHILEV	I	2	1280	WORK10	2560
PI	R	4	12	TSTXEV	48
PIJ	R	4	12	TSTXEV	48
PL	R	4	3	TSTXEV	12
PLSB	I	4	6	TXMAPS	24
PN	R	4	6	TSBND	24
PNI	R	4	6	TSBSNO	24
PNIP1	R	4	6	TSBSNO	24
PNM1	R	4	6	TSBND	24
PNP1	R	4	6	TSBND	24
PNPD	R	4	1	TSDBN	4
PO	R	4	3	WORK11	12
POP	R	4	6	TSPD2	24
POPEN	R	4	3	TSPD2N	12
PPUP1A	I	2	1	PRFBKD	2
PR	R	4	3	TSTXEV	12
PREMP	I	2	1	WORK11	2
PREMPN	I	2	1	TSPD2N	2
PREMPY	I	2	6	TSPD2	12
PRI	I	2	1	ARIN	2
PRI	I	2	1	DECODE	2
PRI	R	8	1	INIT3	8
PRI	I	4	1	OVERID	4
PRIEN	I	2	6	TSPD2	12
PRIENN	I	2	1	TSPD2N	2
PRILFT	I	2	1	NSEDGE	2
PRIRGT	I	2	1	NSEDGE	2
PRIRT	I	4	256	NSTABL	1024
PRVPDA	R	8	1	NSOUT	8
PRVPDA	R	8	1	PROUT	8
PSP	R	4	1	VPMLF	4
PSUM	R	4	2	TSOUT	8
PT	R	4	6	TSTXEV	24

Table C-4: Variable List  
for FRAME3  
(Sheet 27 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
PTEXT	I	2	1280	WORK10	2560
PTLFLG	L	4	1	OPTNS	4
PTXSC	R	4	1350	PRVP	5400
PUP1F	I	2	1	PREDGA	2
PUP2F	I	2	1	PREDGA	2
Q	R	4	1	DECODE	4
R	R	4	1	DECODE	4
RA	R	4	1	VPFADE	4
RAMTEK	I	4	1	RAMOUT	4
RAMTEK	I	4	1	RAMSET	4
RB	R	4	1	FRM1	4
RB	R	4	1	VPFADE	4
RCOL	I	4	6	RAMCOM	24
RDF343	R	4	1	VPLLI	4
REGCX	R	4	1	WORK4	4
REGCY	R	4	1	WORK4	4
REGCZ	R	4	1	WORK4	4
REGLAT	I	4	1	WORK4	4
REGLON	I	4	1	WORK4	4
REGRAD	R	4	1	WORK4	4
RELFAC	I	2	4096	EDREL	8192
RELLPA	I	2	90	PRVP	180
RELLPB	I	2	90	PRVP	180
RELPRI	I	2	256	EDREL	512
RELTPR	I	2	2	PROUT2	4
RF	R	4	1	VPFDC	4
RFG	R	4	1	VPFDC	4
RFGI	R	4	1	VPFDC	4
RFP	R	4	1	VPLLI	4
RFS	R	4	1	VPFDC	4
RFSI	R	4	1	VPFDC	4
RGHT	I	4	1	TB2	4
RI	R	4	50	PTLSET	200
RIB	R	4	30	CXMAP	120
RIB	R	4	30	STPED	120
RIB	R	4	30	STPLT	120
RIL	R	4	1	STPED	4
RIL	R	4	1	STPLT	4
RILL	R	4	400	GEN	1600
RIR	R	4	1	STPED	4
RIR	R	4	1	STPLT	4
RJ	R	4	50	PTLSET	200
RJK	R	4	1	PRAUPD	4
RJL	R	4	90	PRVP	360
RJLL	R	4	512	GEN	2048
RJLP	R	4	512	GEN	2048
RJLPP	R	4	512	EDGORD	2048
RJLS	R	4	1	LUNEDG	4
RJLS	R	4	2	PTLGEN	8
RJM	R	4	18	PRESM	72
RJR	R	4	90	PRVP	360
RJRP	R	4	200	GEN	800
RL	R	4	1	FRM1	4

Table C-4: Variable List  
for FRAME3  
(Sheet 28 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
RLNA	R	4	1	PTLSIT	4
RLNE	R	4	1	STPED	4
RLNE	R	4	1	STPLT	4
RLZE	R	4	1	STPED	4
RLZL	R	4	1	STPLT	4
RNA	R	4	8	LOCAL	32
RNA	R	4	25	LUNEDG	100
RNA	R	4	46	NSOUT	184
RNA	R	4	46	PATPRO	184
RNA	R	4	47	PROUT	188
RNA	R	4	11	SAVELT	44
RNA	R	4	46	VPAINC	184
RNA	R	4	15	VPLTC	60
RND4	R	4	1	EDGGEN	4
ROT	I	2	2	DCOUT	4
RP	R	4	3	FIXDT	12
RPC	R	4	3	FIXDT	12
RPRE	R	4	1	TSTXMD	4
RR	R	4	1	FRM1	4
RSLTN	I	4	6	MODSET	24
RSLTN	I	4	6	SETRD	24
RSR	R	4	1	VPLNDL	4
RT	R	4	1	FRM1	4
SAVSSW	L	4	32	SVSSW	128
SCALE	I	2	60	WORK15	120
SCL	R	4	1	TSTXMD	4
SEM	I	2	18	PRESM	36
SH	R	4	1	EDGGEN	4
SH4	R	4	1	EDGGEN	4
SHK	R	4	8	TSOUT	32
SHRTEX	I	2	1	WORK11	2
SHVDWN	I	2	1	PRFONM	2
SHVUP	I	4	1	PREDGA	4
SINGFL	R	4	1	SINGS	4
SINGLE	I	2	2	DCOUT	4
SIZLT	R	4	400	GEN	1600
SKY	I	2	3	VPFM	6
SL	R	4	1	TSEA	4
SLMAX	R	4	1	PRAREA	4
SLOP	R	4	100	STRIP	400
SLP	I	2	1	EDGGEN	2
SLPSGN	I	4	1	PROUT	4
SMLE	I	2	1	PREDGA	2
SN	R	4	3	FRM1	12
SPCASE	I	4	1	DECODE	4
SR	R	4	1	TSEA	4
SS	I	2	3	TSPD1	6
SSN	I	2	1	TSPD1N	2
SSW	L	4	32	SSWTCH	128
STPFAC	I	2	256	NSEDGE	512
STRFAC	I	2	256	NSEDGE	512
SV	R	4	3	FIXDT	12
SZP	I	4	1	ARECAL	4

Table C-4: Variable List  
for FRAME3  
(Sheet 29 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
T1	I	2	1	PRVIS	2
T2	I	2	1	PRVIS	2
T3	I	2	1	PRVIS	2
T4	I	2	1	PRVIS	2
T5	I	2	1	PRVIS	2
TBF	I	2	1	ARIN	2
TBLCNT	I	2	1	NSEDGE	2
TBLK	I	4	5	MODSET	20
TBLK	I	4	5	PUT	20
TBLK	I	4	5	REED	20
TBLK	I	4	5	SETRD	20
TBUSE2	I	2	1	WORK4	2
TCOLOR	I	2	144	WORK6	288
TDJL	I	4	1	TEXDMP	4
TDJR	I	4	1	TEXDMP	4
TEEFD	I	4	18	PRESM	72
TEEFDA	I	4	1	PRFMD	4
TEEFDB	I	4	1	PRFMD	4
TEX	R	8	1	INIT3	8
TEXCOD	I	2	1	WORK11	2
TEXFLG	L	4	1	OPTNS	4
TL	R	4	3	TSOUT	12
TLAYER	I	4	1920	WORK21	7680
TLFENB	I	4	1	PRFBKD	4
TLFENT	I	4	1	PRFBKD	4
TMP	I	2	256	EDORD	512
TMP1	R	4	1	ORDER	4
TNEFLG	I	4	1	NSOUT	4
TNEFLG	I	4	1	PROUT	4
TNX	R	4	64	WORK17	256
TNY	R	4	64	WORK17	256
TNZ	R	4	64	WORK17	256
TOL	R	4	1	PRESEL	4
TOL	R	4	1	PRTPLU	4
TR	R	4	3	TSOUT	12
TRANS1	C	1	1	WORK4	1
TRIG	I	2	1	COLOR	2
TRILAY	I	2	1	COLOR	2
TRINUM	I	2	1	COLOR	2
TS	I	2	3	PRLD	6
TSCN	R	4	1500	STRIP	6000
TSCND	R	4	360	PRVP	1440
TSFBB	I	2	256	PRFBKM	512
TSFBT	I	2	256	PRFBKM	512
TSFLG	I	2	1	PREDGE	2
TSFOB	I	2	1	PRFONM	2
TSFOT	I	2	1	PRFONM	2
TSK	R	4	8	TSOUT	32
TSM	I	2	18	PRESM	36
TSTFAC	I	2	1	COLOR	2
TSUM	R	4	4	TSOUT	16
TXCODE	I	2	18	TSPD2	36
TXCODN	I	2	3	TSPD2N	6

Table C-4: Variable List  
for FRAME3  
(Sheet 30 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
TXSC	R	4	3000	GEN	12000
TXSHDF	I	2	6	TSPD2	12
TXSHDN	I	2	1	TSPD2N	2
XTAB	I	4	3	MISC	12
JCNT	I	2	1	NSEDGE	2
JFACEL	I	4	200	NSRSLV	800
JFACER	I	4	200	NSRSLV	800
JFOFBB	I	2	1	PRFBKD	2
JFOFBT	I	2	1	PRFBKD	2
JFOP1B	I	2	1	PRFBKD	2
JFOP1T	I	2	1	PRFBKD	2
JFOP2B	I	2	1	PRFBKD	2
JFOP2T	I	2	1	PRFBKD	2
JJLEFT	I	4	200	NSRSLV	800
JLOC	I	4	1	NSRSLV	4
JPRIRT	I	4	200	NSRSLV	800
JSEDT	I	2	8	PRTPL	16
JSEP3	R	4	1	PRDMP	4
JSEP3	R	4	1	PRFBKU	4
JSEP3B	I	2	1	PRP3S	2
JSEP3T	I	2	1	PRP3S	2
JVSWS	R	4	9	FIXDT	36
JVW	R	4	9	FRM1	36
VE	I	2	4	PRLD	8
VFOV	R	4	1	FRM1	4
VIB	I	2	1	PREDGE	2
VIT	I	2	1	PREDGE	2
VLABEL	I	2	256	WORK4	512
VP	R	8	3	FRM1	24
VPN	R	4	9	FRM1	36
VX	R	4	256	WORK4	1024
VY	R	4	256	WORK4	1024
VZ	R	4	256	WORK4	1024
WJ	R	4	2	ARECAL	8
WND	R	4	3	FRM1	12
WNDFLG	L	4	1	JWIN	4
X	R	4	1	PTLSIT	4
X	R	4	1	STPED	4
X	R	4	1	VIDPRO	4
XCD	R	4	1	TSESP	4
XCDJ1	R	4	1	TSESP	4
XCDJL	R	4	1	TSESP	4
XCDL1	R	4	1	TSESP	4
XCDSUM	R	4	2	TSINC	8
XCN	R	4	1	TSESP	4
XCNJ1	R	4	1	TSESP	4
XCNJL	R	4	1	TSESP	4
XCNL1	R	4	1	TSESP	4
XCNSUM	R	4	6	TSINC	24
XJL	I	2	2	TSINC	4
XLATMN	I	2	1	TSPD2N	2
XLATMS	I	2	6	TSPD2	12
XLT	R	8	1	INIT3	8

Table C-4: Variable List  
for FRAME3  
(Sheet 31 of 32)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
XMAP	I	4	256	TXMAPS	1024
XP	R	1	12	TSESP	48
XS	T	2	1	ARIN	2
XSIZ	I	4	1	RAMSET	4
YSIGN	I	4	1	RAMSET	4
YSIZ	I	4	1	RAMSET	4
ZFACOD	I	4	1	COLOR	4
ZPAREN	I	4	1	COLOR	4
ZPRIO	I	4	1	COLOR	4

Table C-4: Variable List  
for FRAME3  
(Sheet 32 of 32)

	TOTAL
SIZE	5,774
DIMN	93,295
TOTAL	263,488

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
AOUPL	I	2	512	PRIUOL	1024
AUORPL	I	2	512	PRIUOL	1024
BUFF	I	4	320	BFRI	1280
BUFF	I	4	320	BFRO	1280
COMNAM	R	8	1	PPINP	8
D.AA	I	4	1	MODCNT	4
D.AA	I	4	1	NEWBLK	4
D.AA	I	4	1	PPCNT	4
D.AA	I	4	1	PPFPL	4
D.AA	I	4	1	PPINP	4
D.AA	I	4	1	PPLIST	4
D.AA	I	4	1	PPUOL	4
D.AA	I	4	1	PUT	4
D.AA	I	4	1	RDBLK	4
D.AB	I	4	1	PPINP	4
D.AB	I	4	1	PPLIST	4
D.BA	I	4	1	MODCNT	4
D.BA	I	4	1	PPCNT	4
D.BA	I	4	1	PPFPL	4
D.BA	I	4	1	PPLIST	4
D.BA	I	4	1	PPUOL	4
DPL	R	4	1	PPCNT	4
EOF	I	4	1	REED	4
FBUF	R	4	4	PPINP	16
FCDAT	I	2	32768	PRI AFL	65536
FCSEQ	I	2	4096	PRI AFL	8192
FDAT	R	4	1	PPCNT	4
FILNAM	R	8	1	PPINP	8
FLAGS	I	4	832	PRI AFL	3328
FNAM	R	8	1	RDBLK	8
FNM	R	8	3	RDBLK	24
FORI	R	4	1	PPINP	4
FORJ	R	4	1	PPINP	4
FPLNAM	R	8	1	WRTFPL	8
FPRI	I	4	4096	CPSTM	16384
HIADR	I	2	1	PPFPL	2
I	I	4	1	MODCNT	4
I	I	4	1	NEWBLK	4
I	I	4	1	PPCNT	4
I	I	4	1	PPFPL	4
I	I	4	1	PPINP	4
I	I	4	1	PPLIST	4
I	I	4	1	PPMSG	4
I	I	4	1	PPSORT	4
I	I	4	1	PPUOL	4
I	I	4	1	PRIPRO	4
I	I	4	1	PUT	4
I	I	4	1	REED	4
IO	I	2	1	MODCNT	2
IO	I	2	1	NEWBLK	2
IO	I	2	1	PPFPL	2
IO	I	2	1	PPINP	2
IO	I	2	1	PPLIST	2

Table C-5: Variable List  
for PRIPRO  
(Sheet 1 of 7)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
-----	-	-	-----	-----	-----
I1	I	2	1	MODCNT	2
I1	I	2	1	PPFPL	2
I1	I	2	1	PPLIST	2
I2	I	2	1	PPCNT	2
I256	I	2	1	NEWBLK	2
I2BUF	I	2	8	PPINP	16
I320	I	2	1	NEWBLK	2
I5	I	2	1	PPFPL	2
I5	I	2	1	PPLIST	2
IABSAD	I	4	1	SETRD	4
IACT	I	4	1	PRIBLK	4
IADR	I	4	1	RDBLK	4
IARG	I	4	1	PUT	4
IARG	I	4	1	REED	4
IARG	I	4	1	SETFIL	4
IARG	I	4	1	SETRD	4
IB	I	4	1	PPLIST	4
IB1	I	4	1	NEWBLK	4
IB2	I	4	1	NEWBLK	4
IB3	I	4	1	NEWBLK	4
IBFAC	I	2	1	PPFPL	2
IBFAC	I	2	1	PPLIST	2
IBIT	I	2	8	PPLIST	16
IBITM	I	2	16	PPLIST	32
IBUF	I	4	4	PPINP	16
IBUO	I	2	1	PPFPL	2
IBUO	I	2	1	PPLIST	2
IBYTE	I	4	1	PPCNT	4
IC	I	4	1	MODCNT	4
ICLOS	I	4	1	PPINP	4
ICODE	I	4	1	PPINP	4
ID	I	4	1	PPLIST	4
IDAT	I	2	8	PPLIST	16
IDAT	I	4	512	PRIBLK	2048
IDATA	I	4	528	PRIBLK	2112
IDBLK	I	2	2	PPINP	4
IDTA	I	4	1	PPCNT	4
IELAP	I	4	1	PRIPRO	4
IERR	I	4	1	PRIPRO	4
IFAC	I	2	1	PPFPL	2
IFAC	I	2	1	PPLIST	2
IFD	I	2	8	PPFPL	16
IFPRI	I	2	4096	CPSTM	8192
IHDR	I	4	1	PRIBLK	4
IHEAD	I	4	16	PRIBLK	64
IHIGH	I	2	1	PPFPL	2
II	I	4	1	MODCNT	4
II	I	4	1	PPFPL	4
II	I	4	1	PPINP	4
II	I	4	1	PPLIST	4
IJ	I	4	1	MODCNT	4
IJ	I	4	1	PPLIST	4
IK	I	4	1	PPINP	4

Table C-5: Variable List  
for PRIPRO  
(Sheet 2 of 7)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
ILAST	I	4	1	NEWBLK	4
ILAY	I	4	1	PPLIST	4
ILOW	I	2	1	PPFPL	2
ILU	I	4	3	RDBLK	12
IM1	I	4	15	PPMSG	60
IM2	I	4	15	PPMSG	60
IM3	I	4	15	PPMSG	60
IM4	I	4	15	PPMSG	60
IM5	I	4	15	PPMSG	60
IM6	I	4	15	PPMSG	60
IM7	I	4	15	PPMSG	60
IM8	I	4	15	PPMSG	60
IMESS	I	4	120	PPMSG	480
IMIN	I	4	1	PRIPRO	4
IMOD	I	2	1	PPFPL	2
IMOD	I	4	1	PPINP	4
IMOD	I	2	1	PPLIST	2
INC	I	4	1	PRIBLK	4
INDX	I	2	8	MODCNT	16
INDX	I	2	32	PPUOL	64
INIT	I	4	3	RDBLK	12
INUM	I	4	1	MODCNT	4
INUM	I	4	4	PPINP	16
IOBJ	I	2	1	PPFPL	2
IOBJ	I	4	1	PPINP	4
IOBJ	I	2	1	PPLIST	2
IP	I	4	2	PPCNT	8
IP	I	4	1	PPUOL	4
IPAIR	I	4	1	PPCNT	4
IPB	I	2	1	PPFPL	2
IPB	I	4	1	PPINP	4
IPB	I	2	1	PPLIST	2
IPD	I	4	1	NEWBLK	4
IPRI	I	4	1	PPFPL	4
IPROC	I	4	1	PUT	4
IPROC	I	4	1	REED	4
IPROC	I	4	1	SETRD	4
IR	I	4	1	PPFPL	4
IR	I	4	1	PPLIST	4
IRANG	I	2	1	PPFPL	2
IRANG	I	2	1	PPLIST	2
IREC	I	4	5	BFRI	20
IREC	I	4	5	BFRO	20
IRFC	I	4	1	PUT	4
IRFC	I	4	1	REED	4
IRFC	I	4	1	SETRD	4
IRGHT	I	4	1	PPCNT	4
IRNG	I	2	4096	PPINP	8192
IRX	I	4	1	SETRD	4
ISEC	I	4	1	PRIPRO	4
ISEC	I	4	1	RDBLK	4
ISIDE	I	4	1	PPCNT	4
ISTAT	I	4	1	PRIPRO	4

Table C-5: Variable List  
for PRIPRO  
(Sheet 3 of 7)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
-----	---	---	-----	-----	-----
ISTAT	I	4	1	PUT	4
ISTAT	I	4	1	REED	4
ISTAT	I	4	1	SETFIL	4
ISTAT	I	4	1	SETRD	4
ISTIM	I	4	3	PRIPRO	12
ISUM	I	4	1	PPCNT	4
ISUM	I	4	1	PPLIST	4
ITIM	I	4	3	PRIPRO	12
ITYB	I	4	1	PPFPL	4
ITYB	I	4	1	PPINP	4
ITYP	I	2	1	PPFPL	2
ITYP	I	2	1	PPLIST	2
IUGB	I	4	1	PPINP	4
IUQB	I	4	1	PPINP	4
IUP	I	4	1	PUT	4
IUP	I	4	1	REED	4
IVP	I	4	1	PPCNT	4
IWORD	I	4	1	NEWBLK	4
IWORD	I	4	1	PPINP	4
IWORD	I	4	1	RDBLK	4
IX	I	4	5	BFRI	20
IX	I	4	5	BFRO	20
IZ	I	4	1	MODCNT	4
IZERO	I	4	1	PPLIST	4
IZERO	I	4	1	PRIBLK	4
J	I	4	1	MODCNT	4
J	I	4	1	PPCNT	4
J	I	4	1	PPINP	4
J	I	4	1	PPLIST	4
J	I	4	1	PPSORT	4
J	I	4	1	RDBLK	4
J1	I	4	1	PUT	4
J1	I	4	1	REED	4
JARG	I	4	1	PUT	4
JARG	I	4	1	REED	4
JARG	I	4	1	SETRD	4
JDEX	I	4	1	MODCNT	4
JDEX	I	4	1	PPSORT	4
JDEX	I	4	1	PPUOL	4
JDEX	I	4	1	RDBLK	4
JJ	I	4	1	PPFPL	4
JJ	I	4	1	PPLIST	4
JJ	I	4	1	PPUOL	4
JJ	I	4	1	RDBLK	4
JP	I	4	1	PPSORT	4
JPREV	I	4	1	MODCNT	4
JREC	I	4	1	REED	4
JX	I	4	1	PUT	4
JX	I	4	1	REED	4
K	I	4	1	PPCNT	4
K	I	4	1	PPSORT	4
K	I	4	1	PUT	4
K	I	4	1	REED	4

Table C-5: Variable List  
for PRIPRO  
(Sheet 4 of 7)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
KARG	I	4	1	PUT	4
KDEX	I	4	1	PPSORT	4
KK	I	4	1	PPUOL	4
KNUM	I	4	1	PPFPL	4
KNUM	I	4	1	PPLIST	4
KVAL	I	4	1	PPSORT	4
LAYER	I	4	1	PPFPL	4
LBLK	I	2	1	PPFPL	2
LBLK	I	2	1	PPLIST	2
LCODE	I	4	3	RDBLK	12
LEFT	I	4	1	PUT	4
LEFT	I	4	1	REED	4
LENG	I	4	1	RDBLK	4
LGCOD	I	4	1	RDBLK	4
LMOD	I	2	1	PPLIST	2
LOADR	I	2	1	PPFPL	2
LOADRP	I	4	1	PPFPL	4
LOBJ	I	2	1	PPLIST	2
LOHI	I	2	8192	PRISRT	16384
LPCT	I	4	1	PUT	4
LPCT	I	4	1	REED	4
LRANG	I	2	1	PPLIST	2
LUAFL	I	4	1	PPINP	4
LUAFL	I	4	1	PRIPRO	4
LUBLK	I	4	1	RDBLK	4
LUERR	I	4	1	PPMSG	4
LUO	I	2	1	PPLIST	2
LUSW	I	4	1	MODCNT	4
LUSW	I	4	1	NEWBLK	4
LUSW	I	4	1	PPFPL	4
LUSW	I	4	1	PPINP	4
LUSW	I	4	1	PPLIST	4
LUSW	I	4	1	PPUOL	4
LUSW	I	4	1	PRIPRO	4
M	I	4	1	PPINP	4
M	I	4	1	PPUOL	4
M	I	4	1	PUT	4
M2	I	4	1	PPSORT	4
MACT	I	4	16	PPCNT	64
MHDR	I	4	1	PPCNT	4
MINRNG	I	4	1	PPINP	4
MJ	I	4	1	PPLIST	4
MN1	I	2	1	PPFPL	2
MN1	I	2	1	PPLIST	2
MNUM	I	2	1	NEWBLK	2
MOBJ	I	4	128	PRIMOC	512
MODC	I	4	8	MODCNT	32
MODK	I	4	8	PRIMOC	32
MODNUM	I	2	832	PRIAML	1664
NOL	I	4	1	PPLIST	4
MOVL	I	2	4096	PRILST	8192
MP	I	4	16	MODCNT	64
MPRN	I	2	8	PPFPL	16

Table C-5: Variable List  
for PRIPRO  
(Sheet 5 of 7)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
MSIZE	I	4	1	PRIBLK	4
N	I	4	1	PUT	4
N1	I	4	1	MODCNT	4
N1	I	4	1	PPUOL	4
NAF	I	4	1	PRIAFL	4
NAO	I	2	832	PRIAML	1664
NF	I	4	1	PPFPL	4
NF	I	4	1	PPLIST	4
NFTY	I	4	5	PPINP	20
NGR	I	4	1	PPFPL	4
NGRP	I	4	1	PRILST	4
NGS	I	4	1	PPUOL	4
NMO	I	4	1	PRIMOC	4
NMOD	I	4	1	PRILST	4
NN	I	4	1	PPUOL	4
NO	I	4	1	RDBLK	4
NOBJ	I	2	8	MODCNT	16
NOM	I	4	1	PPLIST	4
NOM	I	4	1	RDBLK	4
NOSEC	I	4	1	PRIPRO	4
NOSSEC	I	4	1	PRIPRO	4
NOWRD	I	4	1	RDBLK	4
NPAIR	I	4	1	PPCNT	4
NPL	I	4	1	PPCNT	4
NRANG	I	4	1	PPFPL	4
NRASP	I	4	1	PRIBLK	4
NRATF	I	4	1	PRIBLK	4
NRSP	I	4	1	PPCNT	4
NRTF	I	4	1	PPCNT	4
NUM	I	4	1	WRTFPL	4
NUMB	I	4	1	PRIMOC	4
NUMBR	I	4	1	PPUOL	4
NUMOBJ	I	2	1	MODCNT	2
NUO	I	4	1	PPLIST	4
NUO	I	4	1	PPUOL	4
NUOC	I	4	1	PRIUOL	4
NUQG	I	2	512	PRIUOL	1024
NWORD	I	4	3	RDBLK	12
RSLTN	I	4	6	SETRD	24
SECDIR	I	4	1	RDBLK	4
SECUG	I	4	1	RDBLK	4
SSW	L	4	32	SSWTCH	128
TBLK	I	4	5	PUT	20
TBLK	I	4	5	REED	20
TBLK	I	4	5	SETRD	20
TEMP	R	4	1	PPINP	4
UGADR	I	2	64	PRIUF	128
UGSCF	R	4	64	PRIUF	256
UORL	I	4	1	PPFPL	4
UORL	I	4	1	PPLIST	4
UORL	I	4	512	PRIUF	2048
VEC	R	4	984	PRIVP	3936
VPSUM	R	4	1	PPCNT	4

Table C-5: Variable List  
for PRIPRO  
(Sheet 6 of 7)

SYMBOL	T	S	DIMN#	LOCATI	TOTAL#
VPV	R	4	3	PRIBLK	12
XSCALE	R	4	1	PRIBLK	4

Table C-5: Variable List  
for PRIPRO  
(Sheet 7 of 7)

	TOTAL
SIZE	1,170
DIMN	69,682
TOTAL	157,242

SYMBOL	T	S	DIMN	LOCATI	TOTAL
AK	R	4	1	FIXDT	4
ATT	R	4	3	FRM1	12
AZIM	R	4	1	FRM1	4
BLKAMT	I	4	8	DRCTRY	32
BLNFLG	L	4	1	OPTNS	4
BUFF	I	4	320	BFRI	1280
BUFF	I	4	320	BFRO	1280
CLC	I	4	18	FR1D	72
COLOR	R	4	768	TABLS	3072
CSI	I	2	6	FR1D	12
CV	R	4	1	FRM1	4
CW	R	4	1	FRM1	4
D.AA	I	4	1	DRCTRY	4
D.AA	I	4	1	PUT	4
D.AB	I	4	1	DRCTRY	4
D.BA	I	4	1	DRCTRY	4
D.BB	I	4	1	DRCTRY	4
DF1B	R	4	1	FADE	4
DF1T	R	4	1	FADE	4
DF2	R	4	1	FADE	4
DFP	R	4	1	CPFM	4
DIR	I	2	1686	DRCT	3372
EDGFLG	L	4	1	OPTNS	4
ELEV	R	4	1	FRM1	4
EOF	I	4	1	REED	4
ERRMSG	I	4	1	MISC	4
F1RFLG	L	4	1	OPTNS	4
FADFLG	L	4	1	OPTNS	4
FILE	R	8	5	SCGEN	40
FOPG	R	4	1	VPFM	4
FOPS	R	4	1	VPFM	4
FORI	R	4	1	INPUT	4
FORI	R	4	1	SCGEN	4
FORJ	R	4	1	INPUT	4
FORK	R	4	1	INPUT	4
FORKSS	R	4	1	INPUT	4
FR1EDB	I	4	2304	FR1D	9216
FVPG	R	4	1	VPFM	4
FVPS	R	4	1	VPFM	4
FWPG	R	4	1	VPFM	4
FWPS	R	4	1	VPFM	4
GND	I	2	3	VPFM	6
HAZG	I	2	3	VPFM	6
HAZS	I	2	3	VPFM	6
HFOV	R	4	1	FRM1	4
I	I	4	1	DRCTRY	4
I	I	4	1	INPUT	4
I	I	4	1	PUT	4
I	I	4	1	REED	4
I	I	4	1	SCGEN	4
IO	I	4	1	FRM1	4
IABSAD	I	4	1	SETRD	4
IARG	I	4	1	DRCTRY	4

Table C-6: Variable List  
for SCGEN  
(Sheet 1 of 5)

SYMBOL	T	S	DIMN	LOCATI	TOTAL
IARG	I	4	1	PUT	4
IARG	I	4	1	REED	4
IARG	I	4	1	SETFIL	4
IARG	I	4	1	SETRD	4
IBN	I	4	1	DRCTRY	4
ICHAN	I	4	1	OPTNS	4
ICOSYS	I	4	1	MISC	4
ICSI	I	4	1	DRCTRY	4
IEF	I	4	1	MISC	4
IELAP	I	4	1	SCGEN	4
IFOGC	I	4	3	CPFM	12
IFXLOD	I	4	1	OPTNS	4
IGNDC	I	4	3	CPFM	12
IHAZC	I	4	3	CPFM	12
ILOD	I	4	1	DRCTRY	4
IMIN	I	4	1	SCGEN	4
INBN	I	4	1	DRCTRY	4
IPROC	I	4	1	PUT	4
IPROC	I	4	1	REED	4
IPROC	I	4	1	SETRD	4
IPTR	I	4	1	DRCTRY	4
IRC	I	4	1	FR1D	4
IREC	I	4	5	BFRI	20
IREC	I	4	5	BFRO	20
IRFC	I	4	1	PUT	4
IRFC	I	4	1	REED	4
IRFC	I	4	1	SETRD	4
IRN	I	4	1	DRCTRY	4
IRX	I	4	1	SETRD	4
ISCR	I	4	3072	INPUT	12288
ISEC	I	4	1	SCGEN	4
ISKYC	I	4	3	CPFM	12
ISOPT1	I	4	1	INPUT	4
ISOPT2	I	4	1	INPUT	4
ISTAT	I	4	1	PUT	4
ISTAT	I	4	1	REED	4
ISTAT	I	4	1	SCGEN	4
ISTAT	I	4	1	SETFIL	4
ISTAT	I	4	1	SETRD	4
ISTIM	I	4	3	SCGEN	12
IT	I	4	1	DRCTRY	4
ITIM	I	4	3	SCGEN	12
IUP	I	4	1	PUT	4
IUP	I	4	1	REED	4
IX	I	4	5	BFRI	20
IX	I	4	5	BFRO	20
J	I	4	1	DRCTRY	4
J	I	4	1	INPUT	4
JO	I	4	1	FRM1	4
J1	I	4	1	PUT	4
J1	I	4	1	REED	4
JARG	I	4	1	PUT	4
JARG	I	4	1	REED	4

Table C-6: Variable List  
for SCGEN  
(Sheet 2 of 5)

SYMBOL	T	S	DIMN	LOCATI	TOTAL
JARG	I	4	1	SETRD	4
JB	I	4	1	DRCTRY	4
JD	I	4	1	DRCTRY	4
JDIR	I	4	1	DRCTRY	4
JE	I	4	1	DRCTRY	4
JEL	I	4	1	JWIN	4
JER	I	4	1	JWIN	4
JL	I	4	1	DRCTRY	4
JREC	I	4	1	REED	4
JSSW	I	4	1	MISC	4
JX	I	4	1	PUT	4
JX	I	4	1	REED	4
K	I	4	1	DRCTRY	4
K	I	4	1	INPUT	4
K	I	4	1	PUT	4
K	I	4	1	REED	4
K1	I	4	1	DRCTRY	4
KARG	I	4	1	PUT	4
KD	I	4	1	DRCTRY	4
KDPTR	I	4	8	DRCTRY	32
KGND	R	4	1	VPFM	4
KI	R	4	1	FRM1	4
KIJ	R	4	4	FIXDT	16
KJ	R	4	1	FRM1	4
KLOD	I	4	1	FIXDT	4
KLTAB	R	4	16	FIXDT	64
KRASH	I	2	1	VPFM	2
KS	R	4	1	FRM1	4
KSKY	R	4	1	VPFM	4
KSS	I	4	1	INPUT	4
KUVW	R	4	6	FIXDT	24
L	I	4	1	DRCTRY	4
LB	I	4	1	DRCTRY	4
LEFT	I	4	1	PUT	4
LEFT	I	4	1	REED	4
LN	I	4	1	MISC	4
LO	I	4	1	MISC	4
LOCFLG	L	4	1	OPTNS	4
LODCRS	I	2	1	DRCTRY	2
LODFIN	I	2	1	DRCTRY	2
LODMOD	L	4	1	OPTNS	4
LPCT	I	4	1	PUT	4
LPCT	I	4	1	REED	4
LSP	I	4	1	MISC	4
LST	I	4	1	MISC	4
LTPARM	R	4	2816	TABLS	11264
LUCLT	I	4	1	INPUT	4
LUCMN	I	4	1	CMNOUT	4
LUENV	I	4	1	DRCTRY	4
LUHDR	I	4	1	INPUT	4
LUVPF	I	4	1	INPUT	4
M	I	4	1	DRCTRY	4
M	I	4	1	PUT	4

Table C-6: Variable List  
for SCGEN  
(Sheet 3 of 5)

SYMBOL	T	S	DIMN	LOCATI	TOTAL
MAXRNG	R	4	8	MISC	32
MINRNG	R	4	8	MISC	32
MK	R	4	1	FIXDT	4
MMAT	R	4	21	MMDAT	84
MMC	R	4	21	MMDAT	84
MMPOS	L	4	1	OPTNS	4
N	I	4	1	DRCTRY	4
N	I	4	1	PUT	4
NE	I	4	1	FIXDT	4
NEWFLG	L	4	1	DRCTRY	4
NFSUM	I	4	1	FIXDT	4
NL	I	4	1	FIXDT	4
NOEDB	I	4	1	FR1D	4
NOSEC	I	4	1	SCGEN	4
NOSSEC	I	4	1	SCGEN	4
NVP	I	4	9	FIXDT	36
OFF	I	4	8	DRCTRY	32
PTLFLG	L	4	1	OPTNS	4
RB	R	4	1	FRM1	4
RBH	I	4	30	DRCTRY	120
REGCT	I	4	1	FR1D	4
RL	R	4	1	FRM1	4
RP	R	4	3	FIXDT	12
RPC	R	4	3	FIXDT	12
RR	R	4	1	FRM1	4
RSLTN	I	4	6	SETRD	24
RT	R	4	1	FRM1	4
SCR	R	4	3072	INPUT	12288
SKY	I	2	3	VPFM	6
SN	R	4	3	FRM1	12
SSW	L	4	32	SSWTCH	128
SV	R	4	3	FIXDT	12
TBLK	I	4	5	PUT	20
TBLK	I	4	5	REED	20
TBLK	I	4	5	SETRD	20
TCSI	I	2	256	DRCTRY	512
TEXFLG	L	4	1	OPTNS	4
TWO16	R	4	1	INPUT	4
TXTAB	I	4	3	MISC	12
UVSWS	R	4	9	FIXDT	36
UVW	R	4	9	FRM1	36
VFOV	R	4	1	FRM1	4
VP	R	8	3	FRM1	24
VPN	R	4	9	FRM1	36
WND	R	4	3	FRM1	12
WNDFLG	L	4	1	JWIN	4
X	R	4	1	INPUT	4
ZC	R	4	1	FADE	4
ZMIN	R	4	1	FADE	4

Table C-6: Variable List  
for SCGEN  
(Sheet 4 of 5)

	TOTAL
-----	-----
SIZE	820
DIMN	15,098
TOTAL	56,498

Table C-6: Variable List  
for SCGEN  
(Sheet 5 of 5)

Appendix D  
Collected Data on Operations

## Appendix D

The data in this appendix represents the data collected regarding the number and types of operations performed on each type of variable. It is divided into six sections as follows:

- Section 1 - 2 Byte Integer Operations
- Section 2 - 4 Byte Integer Operations
- Section 3 - Logical Operations
- Section 4 - 4 Byte Real Operations
- Section 5 - 8 Byte Real Operations
- Section 6 - Other Operations

The column headings for the first five sections are described below.

MODULE	name of the subroutine or program containing the operations
INTERNAL	name of the internal subroutine contained within the module
A# thru R#	number of operations of each type as follows:
A	+
B	-
C	*
D	/
E	**
F	=
G	Arithmetic IF
H	Logical IF
I	ELSEIF
J	.EQ.
K	.NE.
L	.GT.
M	.LT.
N	.GE.
O	.LE.
P	.AND.
Q	.OR.
R	.NOT.

The column headings for the sixth section are described below.

MODULE	name of the subroutine or program containing the operations
INTERNAL	name of the internal subroutine contained within the module
A# thru T#	number of operations of each type as follows: A GOTO B GOTO ASSIGN C Computed GOTO D DO E DO FOR F DO FOREVER G DO UNTIL H DO WHILE I LEAVE J Procedure Call K Subroutine CALL L READ M WRITE N FORMAT O SELECT CASE P CASE Q ASSIGN R REWIND S RETURN T STOP

Appendix D

Collected Data on Operations

Section 1 - 2 Byte Integer Operations

MODULE	INTERNAL	A#	B#	C#	D#	E#	F#	G#	H#	I#	J#	K#	L#	M#	N#	O#	P#	Q#	R#
AREA1							4		2	6	7								
AREA2							7		12	7	16	4					1		
AREA3							2		6	16	22								
AREA4									15		8	9						2	
ARECAL																			
AREMOD																			
CMNOUT																			
COL							5		8			7							
COLOR		4					6		4		1	2	1			1			
COLOR ALAYERING		1					30		7		1	3		2	2	2	2		
CPBLND							1												
CPFADE			1				6												
CPHITE		1					4												
CSDEF							9		8		2	6							
CXMAP		4	1				9		1			1							
DECODE		127	16	21			158		93	7	45	37	22		3	1	7	1	
DLCAL																			
DRCTRY							7												
EDGCAL							2		1		2						1		
EDGGEN							4		5		5	3							
EDGORD							2												
EDGORD FACVT							2		1		1								
EDWOUT																			
ERRRPT																			
FACCOM							3												
FACOUT																			
FACPR3							13		3		2		2				1		
FACOMP							6												
FEP									2		3						1		
FMOD																			
FRAME1							1		1		1								
FRAME2																			
FRAME3									2		4								
HDROUT																			
INIT2																			
INIT3							4												
INPUT							2												
UNGLAT																			
LED									1			1							
LR2		3					45		20	2	16		6	2			2		
LR2 MODSELECT							2		1	1	2								
LSTOUT																			
MDCLR2																			
MMFAD																			
MODCLR																			
MODCNT							2												
MODFY2																			
MODIFY																			
MODRD																			
MODSET																			
MODST2																			
MODULA		3	2				73		33	1	31	6					1	1	
MOVE																			

MODULE	INTERNAL	A#	B#	C#	D#	E#	F#	G#	H#	I#	J#	K#	L#	M#	N#	O#	P#	Q#	R#
MULT																			
NEWBLK		1	1				1												
NEWED																			
NEWPL																			
NSEDGR							12		2			2							
NSOUT			1				25		3		1	2							
NSRSLV		3	3				14		8		6	2	1					1	
NSRSLV PRINTEDGE																			
ORDER							4												
OVERID		2					14		20		8	12							
PARSEL		1					9		4		2		1	1					
PATPRO																			
PACNT									1					1					
PPFPL		8	6	1	2		21		4	2	4	1	3					1	
PPINP							12				1								
PPLIST		6	2		1		33		9		2	7	1						1
PPMSG																			
PSORT							3												
PPUOL							1												
PRAPLU		1					33		25		16	16						1	1
PRAREA							2		4	4	11	3						5	
PRALPD									11		3	10							1
PRCLR							1												
PRDMP																			
PRDGR		1	1				15												
PRFEFS							6		17		19	21						12	7
PRELDD							34		7		6	2							1
PREPD		1					12		4		4							1	
PRESEL							38	4	29		29	23						21	1
PRFBKU							120		39	10	82	11	8					32	15
PRINIT							21												
PRIPRO							1												
PRIRSV		4					4		3		2	2							
PRNEFS							22		20		9	24						7	6
PRNXTO		3					10		15		16	2	2					3	
PROUT				2			47		14		8	5	3	2				3	
PASTOR							12												
PATPLU		1					23		21		19	15	1		1			8	2
PATPLU CLEARTRAN							3												
PRVIS		13		15			10												
PTCAL							1		1		2							1	
PTCLR2																			
PTLGEN																			
PTLSIT							2		3			3							
PUT																			
PUT2																			
PUTCLR																			
PUTSET																			
PUTST2																			
RAMOUT																			
RAMSET								13											
RDBLK																			
REED																			

MODULE	INTERNAL	A#	B#	C#	D#	E#	F#	G#	H#	I#	J#	K#	L#	M#	N#	O#	P#	Q#	R#
REED2																			
ROTMAT																			
RSTPED																			
SAVELT							3												
SCGEN																			
SETFIL																			
SETRD																			
SETRD2																			
SINGS		3					164		116		114	2	5	3	1		4	7	
SINGS MODSELECT							2		1	1	2								
STPED							4		2		2								
STPLT							5		1		2								
TB2		1					163		62		27	39	1	1			5	2	
TB2 MODSELECT							2		1	1	2								
TMULT																			
TRANS																			
TSBNGT		4					55		23		4	10		2	7	3	1	2	
TSBSNG			2				4		3		1	5		1	1		1	3	
TSDBN			1						1			1							
TSEA																			
TSEDA									1			1							
TSEDGR		1					18		3		1		1		3			2	
TSEMCV			1				32		1			1							
TSESP		3	2				18		4		4		1	1	1			2	
TGINIT							18												
TSLOD																			
TSLODS							21		15		3	7	1	2	1	1			
TSMUX			2						3		2			2				1	
TSPINC		1	1				3		9		9	2							
TSSHAD			2						5			3		2					
TSTXMD			1				6		7		3	5	2						
TSTXMD SETUPLOD																			
TSTXMD SETUPMAP		2					1		1		1		1						
TMUL																			
TVEC																			
UPDATE																			
VEC																			
VIDOUT																			
VIDPRO							3		2		4	1							
VPAINC							30		1			1							
VPCFC							9		1		4								
VPADE							2		1		1								
VPIFLD																			
VPILN							2												
VPANDL																			
VPLTC							5												
VPMLF							14				3								
VPSIMP							15		1		3								
VPTX		2					2												
VTP		1							14	1	11	5						1	
WINDOW																			
WINDMP																			
WRTFPL																			

Appendix D

Collected Data on Operations

Section 2 - 4 Byte Integer Operations

MODULE	INTERNAL	A#	B#	C#	D#	E#	F#	G#	H#	I#	J#	K#	L#	M#	N#	O#	P#	Q#	R#
AREA1							9												
AREA2							13												
AREA3							24												
AREA4							2												
ARECAL							5		3		6						3		
AREMOD																			
CYNOUT																			
COL																			
COLOR							4		2					2			1		
COLOR ALAYERING	1						5		1		1								
CPBLND							1												
CPFADE			1																
CPLITE	1						3		1			1							
CSDEF							2												
CXMAP	5	4					24		15		3	5	1		4	2			
DECODE	9						15												
DLCAL	2						5												
DRCTRY	14	5	2				32		7		7		4	1			5		
EDGAL	12				1		53		7	2	4	1	1	2		1	1	1	
EDGGEN	14	2					40		9	2	10		2	2		1	5	1	
EDGORD		5					14		6				5	3			2		
EDGORD FACVT	1						2		2			1				1			
EDWOUT	4						6		3		1		2						
ERRRPT																			
FACCOM	5	1	6	1			15		8		3	4	1						
FACOUT	5						12				1								
FACPRO	20	4			1		37		9		6		6				2		
FACOMP							1												
FED							51		19		40	1					4	8	
FMOD	9						2		1		1								
FRAME1	10	3	4	1			30		9		4	1	5	1					
FRAME2	28	11	10	2			71		6		1	2	3						
FRAME3	5	3	4	1			16		7				5	3		3		5	
HDROUT	2						4		1	1	2								
INIT2		1					27												
INIT3	1				1		10		2						2	2	2		
INPUT	24	2	1				38												
LNGLAT									1		1								
LDD		7	3	2			8		1				1						
LR2							60		1		1								
LR2 MODSELECT							2												
LSTOUT																			
*DCLR2							2		2			1	1						
*MMFAD							10												
*MODCLR							2		2			1	1						
*MODCNT	6	2	1				18		1				1						
*MODFY2	10	2			1		28		6	1	2	2	2	2				1	
*MODIFY	10	2			1		28		6	1	2	2	2	2				1	
*MODRD	8	3			1		19		5		1	1	2	1		1		1	
*MODSET	3			1	1		5		2			1	1	1				1	
*MODST2	3			2	1		5		2			1	1	1				1	
*MODULA	2						7		9	2	3	5	3			2	1		
*MOVE																			

MODULE	INTERNAL	A#	B#	C#	D#	E#	F#	G#	H#	I#	J#	K#	L#	M#	N#	O#	P#	Q#	R#
MULT																			
NEWBLK		1		1			17		3		2					1			
NEWED		5					25		6		2	2	4			1	1	1	
NEWPL		5					15		1							1			
NSEDGR		1					7		3		2		1			1	1		
NSOUT		7					36		5	1	3	1	6	1					
NSRSLV		17	5				85		20	1	11	4	6	3	1	5	1		
NSRSLV PRINTEDGE		1					2									1			
ORDER		6	2				4		3		3								
OVERID		11					8		4		4								
PARSEL							1												
PATPRO		3	2				5		2				2						
PPCNT		13	2	4	1		30		3		2			1					
PPFPL		1	6	2	1		16		6		1		4						
PPINP		6	1				24		10		5		5	1	1		2	1	
PALIST		6	5	1			25		3		1		2						
PPMSG																			
PASORT		2	2				7		2		1				1				
PPUGL		4	1				10		1		1								
PRAPLU		2					30		14		4	10	6				5	1	
PRAREA									1		1								
PRAUPD			1				1		1		1						1		
PRCLR							1												
PRDMP									2		2								
PREDGR		2					7		6	1	2	2	1			1			
PREEFS							32												
PRELOD		1					5		6		4		2						
PREPD							30	1	23	1	18	15	1	1	2		9	3	
PRESEL			4				27		1		1								
PRFBKU							25		15	3	17	8	5				5	7	
PRINIT		1					12												
PRIPRO		4	1	4	1		6		6		3	1	2						
PRIRSV							4		4		3	2				1	1		
PRNEFS							17												
PRNXTO							12		13		11	2						2	
PROUT		14					60		5	2	2	4	7				1		
PRSTOR							4		1				1						
PRTPLU							59		29	4	11	12	11		3	2	7	3	
PATPLU CLEARTRAN							2												
PRVIS							1												
PTCAL		5					11		2				2						
PTCLR2							1		2				1	1					
PTLGEN		2					9		6		7		1			1			
PTLSIT		1					17		5		1	1	2			1			
PUT		9	2		1		25		5	1	2	1	2	2				1	
PUT2		9	2		1		25		5	1	2	1	2	2				1	
PUTCLR							1		2				1	1					
PUTSET							2												
PUTST2							2												
RAMOUT							6		5	2	5	2							
RAMSET		2	3	1	2		32		6		5				1				
ROBLK		9	1	5	2		21		5		1	2	2						
REED		11	3		1		30		5	2	3	1	2	2				1	

MODULE	INTERNAL	A#	B#	C#	D#	E#	F#	G#	H#	I#	J#	K#	L#	M#	N#	O#	P#	Q#	R#
REED2		11	3		1		30		5	1	2	1	2	2					1
ROTMAT																			
RSTPED		1					1		1			1							
SAVELT		2					18		2			1	1						
SCGEN		4	1	4	1		5												
SETFIL							1		1			1							
SETRD		2		1	1		5		2			1	1	1					1
SETRD2		3		2	1		5		2			1	1	1					1
SINGS							141												
SINGS MODSELECT							2												
STPED		9	3				24		9		4		3	2					
STPLT		8	3				23		10		3	1	3	2	1				
TB2							136												
TB2 MODSELECT							2												
TMULT																			
TRANS																			
TSENST							2		1										
TSBSNO			1				6		2		2						1		
TSDBN							5		1		1								
TSEA		1	1				4		4		2	1	1						
TSEDA		3					7		1		6			2	2				2
TSEDGR		3	1				3		1							1			
TSEMOV		14	1				3												
TSESP		15	3				3		3				1		2	1	2	1	
TSINIT			3	1	1		4		1				1						
TSLOD			1				2		1				1						
TSLODS			2				31												
TSMUX			2	2			2												
TSPINC			5				28		3	2	5						1	1	
TSSHAD			1																
TSTXMD		10	3	1			56		11	2	8	1	3	1		1			4
TSTXMD SETUPLOD		4	1		1		7								1				
TSTXMD SETUPMAP		4	4	1			24		9		7	2	3	1			1		2
TTMUL																			
TVEC																			
UPDATE			1				1												
VEC																			
VIDOUT																			
VIDPRO		8	3	1	3		18		12		7	2	6	2			3	3	
VPAINC		3	3				25		7		1	1	3	2	1		1	1	
VPCFC		1	1				4	1	5		5	3	1						7
VPADE							1		6		5	1							
VPIFLD							1												
VPILN		1					9												
VPLNDL		2	2	2		2			1		1								
VPLTC		5	4	1			26	1	6		2		3	2	1		1	1	
VPLLF		2	2	2	2		9		5		6	1							1
VPSIMP		2	2	1	1		5		3	1	1		1		2	1	2		
VTEX		2	1	1	1		13		5		4				1				
VTO		1	1				10		4		4								
WINDOW					2														
WDDMP		5	4				22		6		2		8						2
WTFPL		1			1		1												

Appendix D  
Collected Data on Operations  
Section 3 - Logical Operations

MODULE	INTERNAL	A#	B#	C#	D#	E#	F#	G#	H#	I#	J#	K#	L#	M#	N#	O#	P#	Q#	R#
AREA1																			
AREA2																			
AREA3																			
AREA4																			
ARECAL																			
AREMOD																			
CMNGUT										1									
COL										1									
COLOR	ALAYERING						2			1								1	
COLOR										1									
CPBLND																			
CPFADE										1								1	
CPLITE																			
CSDEF																			
CXMAP										1									
DECODE																			
DLCAL							1			3									1
DRCTRY							2			5									
EDSCAL							11			4									1
EDGGEN										1								1	
EDGORD							2			3									
EDGORD	FACVT																		
EDWOUT										5									
ERRRPT																			
FACCOM																			
FACOUT							5			6	3						1	1	1
FACPRO							15			23							7	3	5
FADCMP																			
FEP										3							1		
FMOD										3									
FRAME1										11									1
FRAME2							5			9	1								
FRAME3							2			5									
HDROUT																			
INIT2										1									
INIT3							2			4								1	
INPUT							11			2									
LNGLAT																			
LOD										1									
LR2										5								5	
LR2	MODSELECT																		
LSTOUT																			
MDCLR2																			
MMFAD																			
MODCLR																			
MODCNT										2									
MODFY2																			
MODIFY																			
MODRD																			
MODSET																			
MODST2																			
MODULA																			
MOVE																			

MODULE	INTERNAL	A#	B#	C#	D#	E#	F#	G#	H#	I#	J#	K#	L#	M#	N#	O#	P#	Q#	R#
MULT																			
NEWBLK									2										
NEWED							4		9									1	
NEWPL									1										
NSEDGR									1								1		
NSOUT																			
NSRSLV									2								2		
NSRSLV PRINTEDGE									1								1		
ORDER																			
OVERID																			
PASEL									1								1		
PATPRO									2								1		
PCCNT									1										
PPFPL									4										
PPINP									5										
PPLIST									2										
PPMSG									1										
PPSORT									1										
PPJOL									2										
PAPPLU																			
PAREEA																			
PRAUPD																			
PACLR																			
PRDMP																			
PREDGR																			
PREEFS																			
PRELOD																			
PREPD																			
PRESEL																			
PRFBKU																			
PRINIT									2										
PRIPRO									3										
PRIRSV									3								2		
PRNEFS																			
PRNXTO																			
PROUT									1								1		
PRSTOR																			
PRTPLU																			
PRTPLU CLEARTRAN																			
PRVIS									1										
PTCAL									1										
PTCLR2																			
PTLGEN									1										
PTLSIT									1										
PUT																			
PUT2																			
PUTCLR																			
PUTSET																			
PUTST2																			
RAMOUT																			
RAMSET																			
RDBLK																			
REED																			

MODULE	INTERNAL	A#	B#	C#	D#	E#	F#	G#	H#	I#	J#	K#	L#	M#	N#	O#	P#	Q#	R#
REED2																			
ROTMAT																			
RSTPED																			
SAVELT																			
SCGEN																			
SETFIL																			
SETRD																			
SETRD2																			
SING6										68								68	
SING6 MODSELECT										1									
STPED										1									
STPLT										1									
TB2										50								50	
TB2 MODSELECT																			
TMULT																			
TRANS																			
TSBNST																			
TSBSNO																			
TSDBN																			
TSEA																			
TSEDA																			
TSEDGR										1									
TSEMOV																			
TSESP																			
TSINIT																			
TSLOD																			
TSLODS																			
TSMUX																			
TSPINC																			
TSSHAD																			
TSTXMD																			
TSTXMD SETUPLOD																			
TSTXMD SETUPMAP																			
TTMUL																			
TVEC																			
UPDATE																			
VEC																			
VIDOUT																			
VIDPRO										3									
VPAINC																			
VPCFC																			
VPFADE																			
VPIFLD																			
VPILN																			
VPLNDL																			
VPLTC																			
VPMLE																			
VPSIMP																			
VPTEX																			
VTP								9		7									
WINDOW																			
WINDMP																			
WRTFPL										1									

Appendix D

Collected Data on Operations

Section 4 - 4 Byte Real Operations

MODULE	INTERNAL	A#	B#	C#	D#	E#	F#	G#	H#	I#	J#	K#	L#	M#	N#	O#	P#	Q#	R#
AREA1		3					5												
AREA2		1	15				12												
AREA3		8	12				12												
AREA4		4	14	1			10												
ARECAL		6	12	4			16		4		3		1						
AREMOD		8					11		3		2			2				1	
CANOUT																			
COL							1												
COLOR							5												
COLOR ALAYERING																			
CPBLND		1	1	2			7												
CPFADE		1	1	3	1	1	2		2						1	1			
CPLITE		1		1	3		16		5				1	4					
CSDEF																			
CXMAP		5																	
DECODE							3												
DLCAL							1												
DRCTRY																			
EDGCAL			1				25		3	1	1		1	4					
EDGGEN		4	10	7	2		42		7		1	1	3	3	1		2	1	
EDGORD		1		1	1		3		1		1								
EDGORD FACVT							1												
EDWOUT																			
ERRRPT																			
FACCOM							1												
FACGUT		1	1	1		1	6												
FACPRO							11						1		1				
FADCMP		9	12	40	8		33		6				6						
FED		4	23	13	6		145		77		4	3	25	65	33	4	49	8	
FMOD		16	3	34	4		25		1					1					
FRAME1		2	1	7	1	6	23									1			
FRAME2		1	1		5	1	18												
FRAME3																			
FARGUT							2												
INIT2																			
INIT3							7												
INPUT		1	3	1	3		33		1	1			1	1					
LAGLAT			4	4		2	10												
LUD																			
LR2																			
LR2 MODSELECT																			
LSTOUT																			
MDCLR2																			
MMFAD		11	12	23	9		30		9	3			6	1		6			
MODCLR																			
MODCNT																			
MODFY2																			
MODIFY																			
MODRD																			
MODSET																			
MODST2																			
MODULA									1				1						
MOVE							1												

MODULE	INTERNAL	A#	B#	C#	D#	E#	F#	G#	H#	I#	J#	K#	L#	M#	N#	O#	P#	Q#	R#
MULT		1		1				2											
NEWBLK								3											
NEWED								7	1					1					
NEWPL								3											
NSDGR																			
NSOUT								32	6		2		2	3			2		
NSRSLV			1					4											
NSRSLV PRINTEDGE																			
ORDER								4									1		
OVERID																			
PARSEL								1											
PATPRO		12		12				9											
PACNT		2		2				5	1					1					
PPFPL																			
PPINP		2		3				3	1			3						2	
PPLIST																			
PPMSG																			
PPSORT																			
PPUGL																			
PRAPLU									1	4	6							4	
PRAREA		2	16	11	1	2	38		5	1	2	2	2						
PRAPUD		3	6	1			20		4				2	2					
PRCLR																			
PRDMP																			
PRDGR					2			5											
PREEFS																			
PRELDD								3											
PREPD																			
PRESEL			8				17		15				8	8					
PRFBKU			8				25		6		1					5			
PRINIT							1												
PRIPRO																			
PRIRSV																			
PRNEFS																			
PRNXTO																			
PROUT							28		4				2	2					
PRSTOR							5												
PRTPUJ				1			6		2		1		1				1		
PRTPUJ CLEARTRAN																			
PRVIS																			
PTCAL		1			2			3											
PTCLR2																			
PTLGEN		2					6		5		1		1	1	1	1	2		
PTLSIT		3	3				11		4		1		2	1					
PUT																			
PUT2																			
PUTCLR																			
PUTSET																			
PUTST2																			
RAMOUT																			
RAMSET																			
RDBLK																			
REED																			

MODULE	INTERNAL	A#	B#	C#	D#	E#	F#	G#	H#	I#	J#	K#	L#	M#	N#	O#	P#	Q#	R#
REED2																			
ROTHAT		3	3	19			18												
RSTPED							2												
SAVELT							6		2						2				
SCGEN																			
SETFIL																			
SETRD																			
SETRD2																			
SINGS			3				17												
SINGS MODSELECT																			
STPED							13		2					1	1	1		1	
STPLT		2					9		1					1	1			1	
TB2							3												
TB2 MODSELECT																			
TMLT		2					2												
TRANS							1												
TS3NST																			
TSBSNO		5		2	1		12												
TSDBN		5	2	2	2		12												
TSEA		6	13	13	5	2	38		9	1	2	1	2	1	3	1			
TSEDA			25		12		37		1						1				
TSEDGR							6												
TSEMOV							11		1		1								
TSESP		12		5	4		13												
TSINIT							16												
TSLOD			1	2	1		3		1						1				
TSLODS		1		1			31												
TSMUX							3												
TSPINC		3		3			12												
TSSHAD		15		3			11		2				1	1					
TSTXMD		5	4	7	5		18												
TSTXMD SETUPLOD		1					2		2						2				
TSTXMD SETUPMAP																			
TTMUL		1		1			2												
TVEC		1		1			2												
UPDATE																			
VEC		1		1			2												
VIDOUT																			
VIDPRO			2		2		2		2		1			2			2	1	
VPAINC		2	4				75		6				6						
VPCFC			6	10			8												
VPFAD		4		7			25		2					1		1			
VPIFLD		2		2			2												
VPILN		2					9		1				1						
VPLNDL				2			4		1					1					
VPLTC							9												
VPMLF		1	1	4			2		8		8	2		2			6	2	
VPSIMP		2					3		4	1		1	6				3		
VPTX																			
VTP		35	18	69	13	15	95		31	1		2	9	12	4	2	1	2	
WINDOW		4	2	2	8		27												
WINDOWP																			
WRTFPL																			

Appendix D  
Collected Data on Operations  
Section 5 - 8 Byte Real Operations

MODULE	INTERNAL	A#	B#	C#	D#	E#	F#	G#	H#	I#	J#	K#	L#	M#	N#	O#	P#	Q#	R#
AREA1																			
AREA2																			
AREA3																			
AREA4																			
ARECAL																			
AREMOD																			
CMNOUT																			
COL																			
COLOR																			
COLOR ALAYERING																			
CPBLND																			
CPFADE																			
CPITE																			
CSDEF																			
CXMAP																			
DECODE																			
DLCAL																			
DRCTRY																			
EDGCAL																			
EDGEN																			
EDGORD																			
EDGORD FACVT																			
EDWOUT																			
ERRAPT																			
FACCOM																			
FACOUT																			
FACPRG																			
FACOMP		1	4	1	3		5		1	1						2			
FEP																			
FMOD																			
FRAME1		3	3	6	5	4	4												
FRAME2																			
FRAME3																			
-DRCUT																			
INIT2																			
INIT3																			
INPUT											1								
LNGLAT		5	3	3	3	4	7												
LOD																			
LR2																			
LR2 MODSELECT																			
LSTOUT																			
MODLR2																			
MODFAD			9	3	7				1	1						2			
MODCLR																			
MODCNT																			
MODFY2																			
MODIFY																			
MODRD																			
MODSET																			
MODST2																			
MODULA																			
MOVE																			

MODULE INTERNAL A# B# C# D# E# F# G# H# I# J# K# L# M# N# O# P# Q# R#

---

MULT  
 NEWBLK  
 NEWED  
 NEWPL  
 NSEDGR  
 NSOUT  
 NSRSLV  
 NSRSLV PRINTEDGE  
 ORDER  
 OVERID  
 PARSEL  
 PATPRO  
 PCONT  
 PCFOL  
 PCIND  
 PCLIST  
 PPMSS  
 PSORT  
 PUOL  
 PRAPLU  
 PRAREA  
 PRAUPD  
 PRCLR  
 PRDMP  
 PREDGR  
 PREEFS  
 PRELGD  
 PREPD  
 PRESEL  
 PREX  
 PRINT  
 PRIERO  
 PRIERSV  
 PRNEFS  
 PRNXTO  
 PROUT  
 PRSTOR  
 PRTPLU  
 PRTPLU CLEARTRAN  
 PRVIS  
 PTCAL  
 PTCLR2  
 PTLGEN  
 PTLSTT  
 PUT  
 PUT2  
 PUTCLR  
 PUTSET  
 PUTST2  
 RAMOUT  
 RAMSET  
 RDBLK  
 REED

[illegible]

Appendix D  
Collected Data on Operations  
Section 6 - Other Operations

MODULE	INTERNAL	A*	B*	C*	D*	E*	F*	G*	H*	I*	J*	K*	L*	M*	N*	O*	P*	Q*	R*	S*	T*
AREA1		3										5	2								1
AREP2		16										6	2			1					1
AREP3		18										12	2								2
AREP4		14										2	2								1
ARECAL		10																			1
AREMOD		3																			1
CMDOUT												15		1	1						
COL		8										6									2
COLOR												1									1
COLOR PLAYERING												4			5	2					
CPFADE		5																			1
CPLITE		4			3																1
CPLITE		5			1							1									2
CPUSE		8																			9
CPUSE		12										2	2		11	6					
DECODE		161			2													15			
DLCAL					2							1	3			2					1
DRCTRY					12							14	5		7	7					1
EDGCAL						5									3	3	1	4			1
EDGGEN					1	2						12	4		8	8					
EDGORD						7									6	6					
EDGORD FACVT															1	1					
EDWOUT												5									1
EDWOUT																					1
EDWOUT		5			4							3	17		4	4					
FACOUT						1						14	1		1	1					1
FACPRO						7							21		5	9					
FADCOMP					7																
FEP		58										29				5					4
FEP												3									1
FRAME1					12				1	2	2	22	65	2	21	13					1
FRAME2					18							34	38		11	9					1
FRAME3		11			1								19	2	5	5					
HDROUT						1															
INIT2						2							15			1					1
INIT3													24			3					1
INPUT												30	19		1	1					1
LNGLA					2							23									1
LDD					2								1		1	1					
LR2													1		5	1					
LR2 MODSELECT																					
LSTOUT													1								1
MODCLR2												2	3						1	1	
MODCLR					10																1
MODCLR												2	3						1	1	
MODCNT					5								2		4	4					
MODFY2						5						1	7		1	1					
MODIFY												4	7								3
MODRD												2	4			1					3
MODSET												3	4		1	1			1	1	
MODST2												3	4		1				1	1	
MODULA		26								2		1									
MOVE						2															

MODULE	INTERNAL	A#	B#	C#	D#	E#	F#	G#	H#	I#	J#	K#	L#	M#	N#	O#	P#	Q#	R#	S#	T#
MULT					3																1
NEWBLK					3						3	2		6	6						
NEWED		1				2					11	9		1	1						
NEWPL						1					22	1									1
NSEDGR		3									8			1	1						1
NSCUT		3						4			27	3									1
NSRSLV		12				4		1	5		1	6		4	4						5
NSRSLV PRINTED														3	3						
ORDER		1			3																
OVERID		32	1		1						4							3			2
PARSEL		6				1						5		1	3						1
PATPRO						1					4	6									1
PDONT					4						14			1	1						3
PDOPD		1			3							2		6	6						1
PDIND		1			6	2					1	9		14	14						1
POLIST					9						3			12	10						2
PPMSG														2	2						1
PPSORT		2			1									1	1						2
PPSOL					4							1		4	4						2
PRAPPL		13			3						12										
PRAREA																					
PRAPPD		13									4										1
PRCLR																					1
PRDMP		1												19	11						3
PRDPR		2									25										2
PRDPRG		25									14										1
PRELDD		3			2	1						5									1
PREPD		12									2										1
PRESEL		52									10										1
PRFBU											3										1
PRFBU					3	1															2
PRFBU		1									1	10		6	5						1
PRIRSV		5						1				20		1	1						1
PRNEFS		14																			3
PRNXTO		30									8										4
PROUT		18						7			25	3		1	1						1
PROTOR																					2
PRTPLO		17				10						4									2
PRTPLO CLEAR																					
PRVIS														35	36						1
PTCAL					5						4	5		1	1						1
PTCLR2											2	3						1			1
PTLGEN		13			1							9		7	1						3
PTLSIT		7			1							4		7	7						2
PUT					6						2	4		1	1						3
PUT2						6					2	4		1	1						3
PUTCLR											2	2							1		1
PUTSET					1																1
PUTST2						1															1
RAMCUT		6			3						2	7		3	3						1
RAMSET		7			3						14	14		4	4	2	6				1
RDBLK					2						5	8									1
REED					5						2	4		2	2						4

MODULE	INTERNAL	A#	B#	C#	D#	E#	F#	G#	H#	I#	J#	K#	L#	M#	N#	O#	P#	Q#	R#	S#	T#
REED2						5					2	4		1	1						1
ROTMAT											6										1
RSTPED		1																			
SAVELT		2									5	3		1	1						2
SOSEN						2					1	7		2	2						1
SETFIL												2		1	1						
SETRD											3	4		1	1						
SETRD2											3	4		1	1				1	1	
SINGS	167										13	33		33	5						12
SINGS	MODSELECT																				
STPED		18				4					2	8		3	2						2
STPLT		19				3					2	8		1	1						2
TB2		134	3								10	32		53	2	1	2	26			1
TB2	MODSELECT																				
TMULT						3															1
TRANS						2															1
TGBNST		25																			1
TGBND						2	4					1									1
TGBN						1	6				1										1
TSEA		17		1		4	1				4										1
TSEDA		15				3	2				12	5									1
TSEDGR						5					15										1
TSEMOV						2	4				1										1
TSESP		1					5					10		20	13						1
TSEINT						3	5					1									1
TSLCD		2									2										1
TSLCDS		15				13	4				4	1									1
TSMUX						4															1
TSPINC						12					2					1	6				1
TSEHAD		8				3	2				1										1
TSTXMD		2				10					6	6									1
TSTXMD	SETUPALD														2	2					
TSTXMD	SETUPMAP					4			2			2		1	1						
TMUL						3															1
TVEC						2															1
UPDATE												6									1
VED						2															1
VIGOUT												3									1
VIGPRO		3				2	2				7	11		3	3						1
VPAINC		13				4					1	3									1
VPCPD		4				5					4										1
VFADE		7				1					4										1
VPIFLD																					1
VPILN																					
VLNOL																					2
VLTC		4				3					5	4		1	1						5
VMMLF		10				2					11										2
VPSIMP						1	12				6										1
VPTX							2					4									1
VTP		65				1					15			20	20						2
WINDOW											3										1
WDDMP						5			2	1		3		5	5						2
WTFPL												4		1	1						1

~~UNCLASSIFIED~~

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved  
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION / AVAILABILITY OF REPORT  UNLIMITED		
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S)  AFIT/GE/ENG/89D-40			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION  SCHOOL OF ENGINEERING		6b. OFFICE SYMBOL (If applicable)		7a. NAME OF MONITORING ORGANIZATION	
6c. ADDRESS (City, State, and ZIP Code)  AIR FORCE INSTITUTE OF TECHNOLOGY WRIGHT PATTERSON AFB OH 45433-6503			7b. ADDRESS (City, State, and ZIP Code)		
8a. NAME OF FUNDING / SPONSORING ORGANIZATION		8b. OFFICE SYMBOL (If applicable)		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code)			10. SOURCE OF FUNDING NUMBERS		
			PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.
			WORK UNIT ACCESSION NO.		
11. TITLE (Include Security Classification)  DYNAMIC ARCHITECTURE COMPUTER					
12. PERSONAL AUTHOR(S)  PATRICK E PRICE					
13a. TYPE OF REPORT MASTERS THESIS		13b. TIME COVERED FROM _____ TO _____		14. DATE OF REPORT (Year, Month, Day) 1988 DECEMBER	
				15. PAGE COUNT 241	
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	COMPUTER		
			ARCHITECTURE		
			DYNAMIC		
19. ABSTRACT (Continue on reverse if necessary and identify by block number)  THE PURPOSE OF THIS THESIS WAS TO DESIGN A COMPUTER THAT COULD PROCESS A LARGE VARIETY OF CALCULATIONS WITH A MINIMUM OF HARDWARE. THIS CONSTRAINT REQUIRES A COMPUTER THAT CAN CHANGE ITS STRUCTURE TO MATCH THE DEMANDS OF THE PROBLEM CURRENTLY BEING CALCULATED. COMPUTER IMAGE GENERATION WAS SELECTED AS AN EXAMPLE PROBLEM. THE PROCESSING REQUIREMENTS OF REAL-TIME COMPUTER IMAGE GENERATION REQUIRE CALCULATION OF VERY LARGE REAL NUMBERS AS WELL AS VERY SMALL LOGICAL VARIABLES. THE RESULTS DEMONSTRATE THAT, IN A BEST CASE ANALYSIS, A DYNAMIC ARCHITECTURE COMPUTER CAN DEMONSTRATE AN IMPROVEMENT IN PROCESSING SPEED OVER CONVENTIONAL SINGLE INSTRUCTION, SINGLE DATA COMPUTERS.					
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL  PATRICK E PRICE			22b. TELEPHONE (Include Area Code) (513) 255-8926		22c. OFFICE SYMBOL ASD/YWSE

UNCLASSIFIED